Dolog AKF125 → A120/A250
Type: AKF125EN
Version: 7.10
AKF125 for Beginners
User Instruction

DOK–702083.35–1096

Translation of the German Description
DOK–700567.35–0196

Accompanying software package E-No. 424-275182

Documents in the software package

Kit 1

Documentation	**Area of application**
Installation User Instruction DOK-702082	Explains the usage and installation of the dis-kette s included.
How do you proceed? User Instruction DOK-702084	Serves as a ”red thread” through the docu-mentation of the software packet and should be gone over before the start.

Kit 2

Documentation	**Area of application**
AKF125 for Beginners User Instruction DOK-702083	Serves to introduce new customers to AKF125. The user learns how to use the soft-ware in samll steps.
Short Form Guide A120 User Instruction DOK-702087	Tables for validity ranges and symstem mar-kers, SFB-Formal operands for quick use on-site.
Sort Form Guide A250 User Instruction DOK-702088	Tables for validity ranges and symstem mar-kers, SFB-Formal operands for quick use on-site.
Configuration A120 User Instruction DOK-702085	Contains the new features of the current ver-sion and explains th efunctions of th indivi-dual software menus for the configurer.

Kit 3

Documentation	Area of application
Configuration A250 (Vo1) User Instruction DOK-702086	Contains the new features of the current version and explains th efunctions of th individual software menus for the configurer.
Configuration A250 (Vo2) User Instruction DOK-707695	The explanation of the individual software menus will continued.
Masterindex User Instruction DOK-702089	Index of all documentation.

Notes

Application Note

⚠ **Caution   The relevant regulations must be observed for control applications involving safety requirements.**
For reasons of safety and to ensure compliance with documented system data, repairs to components should be performed only by the manufacturer.

Training

Schneider Automation GmbH offers suitable training that provides further information concerning the system (see addresses).

Data, Illustrations, Alterations

Data and illustration are not binding. We reserve the right to alter our products in line with our policy of continuous product development. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us by using the form on the last page of this publication.

Addresses

The addresses are given at the end of this publication.

Terminology

☞ **Note** This symbol emphasizes very important facts.

⚠ **Caution This symbol refers to frequently appearing error sources.**

Warning This symbol points to sources of danger that may cause financial and health damages or may have other aggravating consequences.

Expert This symbol is used when a more detailed information is given, which is intended exclusively for experts (special training required). Skipping this information does not interfere with understanding the publication and does not restrict standard application of the product.

Path This symbol identifies the use of paths in software menus.

Figures are given in the spelling corresponding to international practice and approved by SI (Système International d‘ Unités).
I.e. a space between the thousands and the usage of a decimal point
(e.g.: 12 345.67).

Abbreviatons

ABS	absolute Adressing
Adr.	Adresse (signal adresse)
AE	Block for one time actions
AZ	Block for cyclicel actions
AKF	Instructionlist, Contaktplan, Functionlist
ALD	Sequentiel Flow Chart with Diagnostics
ALS	Sequentiel Flow Chart
AWL	Instructionlist
AWP	User programm
BGT	Subrack
BSS	Serviceconnection for PC ore VS210
DAE	Diagnostics Block for one time actions (AE)
DAZ	Diagnostics Block for cyclicel actions (AZ)
DIB	Diagnostics Block for Block independent Diagnostics
DB0....9	SYM/KOM-Datablock for A120
DPB	Diagnostics Block for Programm Blocks (PB)
DSB	Diagnostics Structure Block
DW	Double word
I/O	Input– / Output signales (e.g. from a Module)
FB	Function block
FUP	Function list
FW	Floatingpoint
HW	Hardware (z.B. PLC)
IB	Initial value block
KB	Sequentiel block
KF	Sequentiel errorbit
KFW	Sequentiel errorword
KOP	Kontaktplan
KS	Kettenstatus für Simultankette
LZS	Run Time System
MW	Markerword
OB	Organisations Block
PB	Program Block
PaDT	Programming- and Diagnostics testequipment
RK	Controlloop
SFB	Standard-Function Block
SK	Step marker

SM	Systemmarker
SSP	Signal memory
SW	Software
SYM	symbolic Adressing
SYM/COM	Symbol und Comment
SZ	Step Counter
TB	Transitionsblock
TN	Teilnehmer
VBGT	Virtuelle Subrack (InterBus-S, Modnet 1/IS))
ZVT	Time organiasationtable (Controlle)
ZZ	Time Counter
<Return>	Applay the key Return
<Esc>	Applay the key Esc
<Ctrl>+<Alt>+<	Applay in the same time the keys Ctrl, Alt und Del (beginning with Ctrl.and finnishing with Del)

Objectives

To provide a general introduction to structured programming.

Arrangement of this guide

Chapter 1	The chapter gives a short overview of the existing components of programming with Dolog AKF. In addition, the basic functions of the software are given.
Chapter 2	Following a general explanation, the various block types in Dolog AKF are described. The different structure levels are demonstrated by means of diagrams.
Chapter 3	The chapter gives a brief insight into the various special languages.
Chapter 4	This chapter describes the most important functions and features of the various block types.
Chapter 5	This chapter includes an example of a small AKF application, complete in all details, for programming the A120 / ALU 200, 201, 202 with AKF125
Chapter 6	This chapter includes an example of a small AKF application, complete in all details, for programming the A250.
Chapter 7	This chapter includes an example of a small AKF application, complete in all details, for programming the A120 / ALU 204, 205 with AKF125.
Chapter 8	This chapter contains several examples of indirect addressing with pointers.

Related Documents

A250
User Manual A250
804 BHB 000 00

A250
User Manual A250
Regeln mit Dolog AKF
804 BHB 001 00

A250
User Manual A250
Prozessperipherie Frontanschlusstechnik
899 BHB 000 00

A250
User Manual A250
Cable
899 BHB 001 00

A250
Blockbibliothek Standard Funktionblocks A250
Vol. 1 (AKF125 V4.x, ALD25 V4.1)
804 BSB 001 00

A250
Blockbibliothek Standard Funktionblocks A250
Vol. 2 (AKF125 V4.x, ALD25 V4.1)
804 BSB 002 00

Validity Note

These User Instructions apply to the AKF125 software, version 7.1, on the DOS operating system.

The current intention is for **remote control data** only to be edited with the **AKF125** configuration software and **not** with ALD25. Therefore, within systems U250 and UZ250, remote control modules **KOS140**, **KOS141** and **DEZ161** should **not** be used as REAL–TIME variants.
Correspondingly, the **KOS 20x** modules of the U120, Z120 and UZ120 systems are **not** to be used.

Table of Contents

Chapter 1 Introduction

This chapter gives you a brief overview of the components available for programming with Dolog AKF. In addition, the basic functions of the software are given.

1.1 Introduction

The Dolog AKF software is used for structured programming of PLC user programs using modern window technology.

For this, programs are created and displayed in three special languages, Instruction List, Ladder Diagram and Function Block Diagram (in this regard see Draft Standard IEC 65A(SEC)65).

Programs consist of different types of blocks which can be connected with each other, depending on the aim of the application and the complexity of the job. In the process, blocks are assigned various tasks (see also Chapter 4):

- Organization of the overall program
- Summary of technical program parts
- Support for repetitive programs (”subroutines”)
- Simplified programming with predefined part programs
- Support for symbolic programming

Following a brief introduction to ”structured programming” with its program components, later chapters go into more detail about ALD25. A brief overview of features is followed by an example in which the first programming steps are practised.

1.2 Programming components

What do you need for programming your programmable controller?

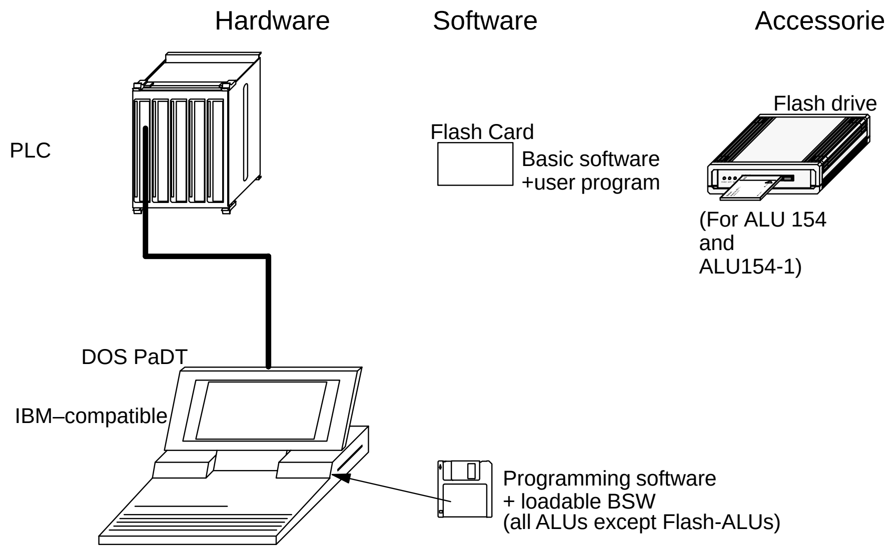


Figure 1 Prerequisites for programming a PLC

1.3 Basic functions

In Dolog, user programs are created off–line and then loaded into the PLC. After that, on–line functions such as an overview of a program during a scan are also possible.

The following basic functions are performed with AKF125:

❐ **Edit** (creation / amendment)	-	off-line/on-line
❐ **Load** (to / from PLC)	-	on-line
Compare (PaDT / PLC)	-	on-line
❐ **Online**	-	on-line
❐ **Print**	-	off-line
❐ **Special**	-	off-line
❐ **Setup**	-	off-line

Chapter 2
Structured Programming

Following a general explanation, the various block types in Dolog AKF125 are described. The different structure levels are demonstrated by means of diagrams.

2.1 Introduction

The performance and economy of a programming system are marked by a number of special features.

Structuring and standardization facilities, the use of general–purpose personal computers as programming units (PaDT) and a convenient user interface add up to advantages that keep software and startup costs as low as possible.

With the present–day mass of information and the volume of programs on programmable logic controllers, program partitioning enables scan times to be optimized. Time critical operations require speedy reactions. This is achieved by means of skillful configuration, by skipping those parts of the program which do not need immediate attention.

The division of a complex of tasks into smaller parts helps to make the bigger picture understandable.

Program parts as individual, closed software blocks are easier to produce and test. Processes integrated into large monolithic main programs, on the other hand, are considerably more difficult to understand in their entirety.

Software configuration with the convenient programming software is useful for avoiding large complexes of branch instructions, which are difficult to manage.

2.2 Program structure

The special languages facilitate the structuring and writing of programs. Programs can be entered and displayed in instruction list, ladder diagram, function block diagram and sequence flow control.

By ”structuring”, we mean the creation of clear, readily understandable, and complete user–program parts, known as blocks.

Processes that are specific to one technical application, as well as processes that are repeatedly used, can be created and tested, then used many times within a system or as a technical block. Function blocks can be arranged together to form universal as well as user–specific program libraries. Standard function blocks, providing complex functions for control, data handling and operation, are integrated into the PLC and form a basis for the simple construction of complex application–specific software blocks.

Blocks are composed of networks. These form the lowest structuring level. The networks contain the logic, which is made up of operations with parameters (known as ”instructions” in the context of instruction list).

The following figures show examples of the various structure levels.

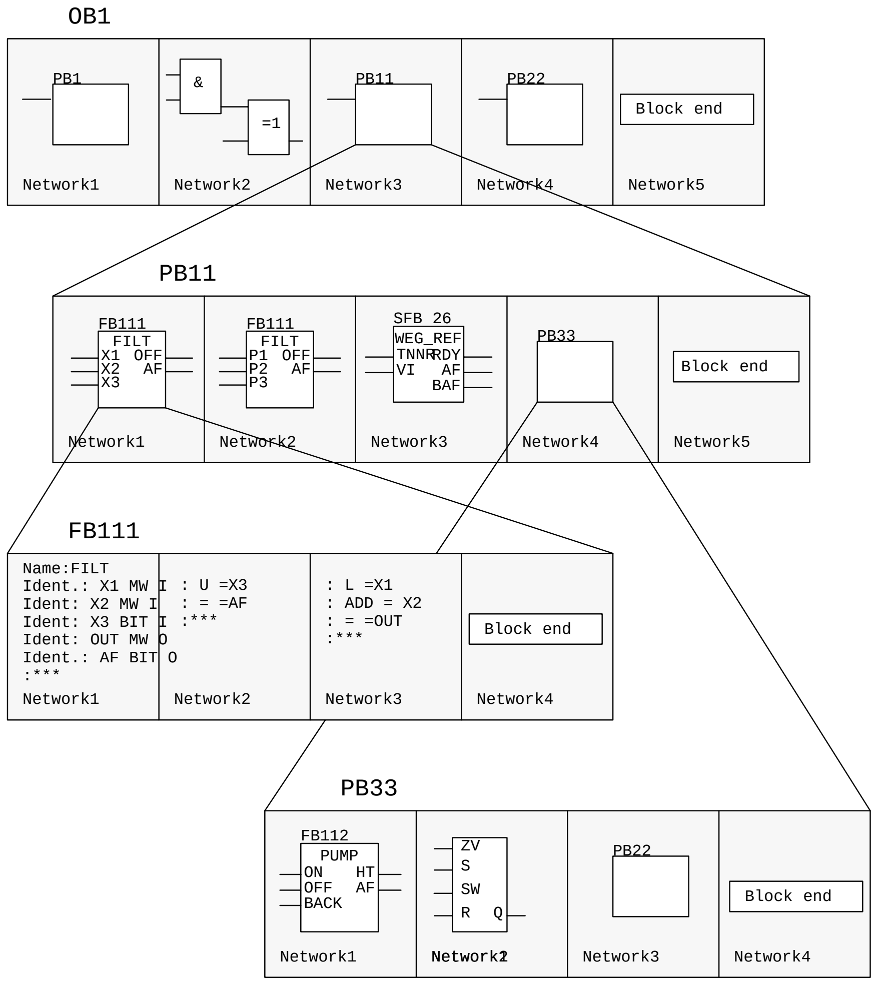


Figure 2 Sample layout of a structured AKF program

Chapter 3
Technical Languages in Programming

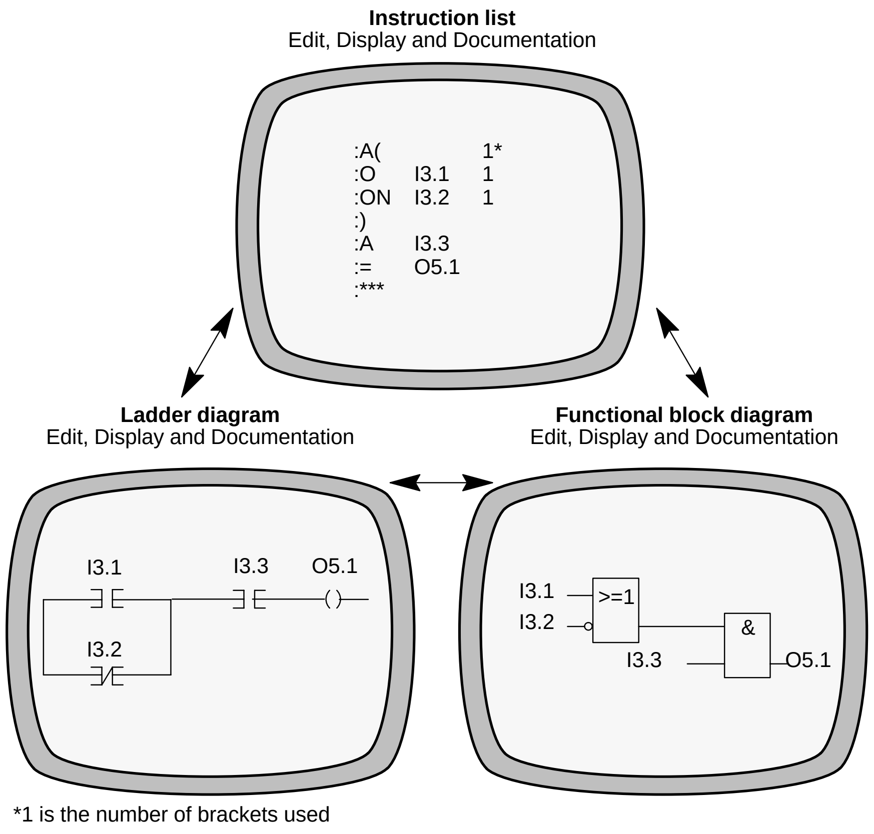
This chapter gives a short explanation of each special language

- ❐ Instruction list IL
- ❐ Ladder diagram LD
- ❐ Functional block diagram FBD

This Dolog software is based on a system of structured programming in the standard special languages. For the standardized definitions (structure etc.) please refer to DIN 19239 or draft standard IEC 65A(SEC)65.

- ❐ Instruction list (IL)
- ❐ Ladder diagram (LD)
- ❐ Functional block diagram (FBD)

Blocks which have been programmed in one particular special language can be displayed in another special language.



Instruction list
Edit, Display and Documentation
:A(1*
:O I3.1 1
:ON I3.2 1
:)
:A I3.3
:= O5.1
:***
Ladder diagram
Edit, Display and Documentation
I3.1
I3.3
O5.1
I3.2
Functional block diagram
Edit, Display and Documentation
I3.1
>=1
I3.2
&
I3.3
O5.1
*1 is the number of brackets used

Figure 3 Display in IL, LD and FBD

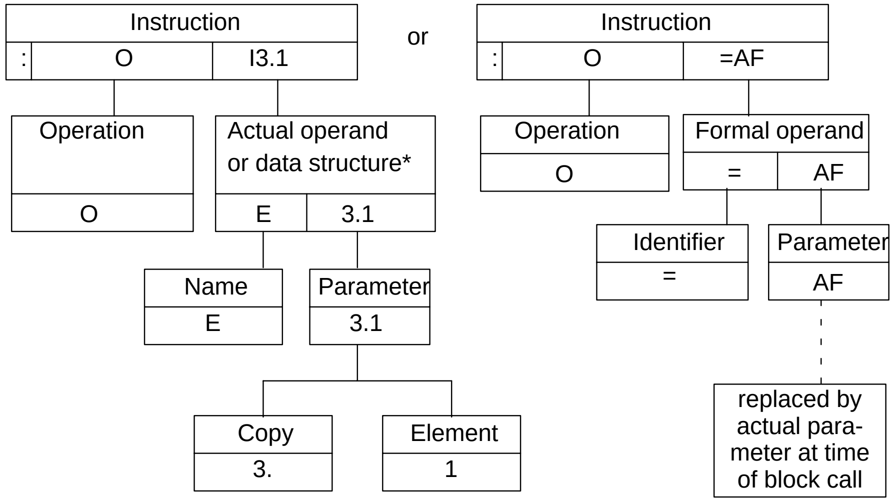
3.1 Instruction list IL

The instruction list is a standard type of presentation in alphanumeric form.

Programs are generated in IL as a sequence of instruction lines.

There are two different types of instructions. The following two instructions are equivalent (AF is arbitrary):

: O I3.1   : O =AF



* Data structure for A250, see chapter 4.6 of AKF125 for beginners

Figure 4 Explanation of terms in Instruction List

A network instruction list closes with the sign for a network end ”***”.
The end of a block is highlighted with ”BE” for block end.

For the individual control systems there are tables of operations and operands/ data structures (data structures for the A250 application).

Organization blocks (OBs), program blocks (PBs) and function blocks (FBs) can be programmed in IL.

Branch and block call instructions are possible in instruction list programming.

☞ **Note:** More precise information, e.g. on creating programs in instruction list, is included in the user instructions ”A120 or A250 configuration”.

3.2 Ladder diagram LD

The ladder diagram is a standard type of graphic presentation.

The following basic symbols are available when generating ladder diagrams.

—] [—	open contact
—]/[—	closed contact
———|	Connection of parallel conductors
———	Continuation in parallel, but without any contact
—( )—	Output

Ladder diagram operations are parameterized with operands/data structures (for data structures see also chapter 4.6).

For explanation of terms, see Figure 4.

The end of a block is marked by a box with ”Block end”.

OBs, PBs and FBs can be programmed in ladder diagram.

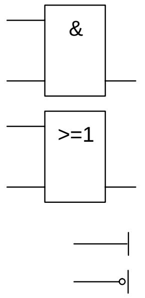
Branch instructions are not permitted in ladder diagram, but block calls are.

☞ **Note:** More detailed information, e.g. on creating ladder diagram programs, is contained in the user instructions ”Configuration”.

3.3 Function block diagram FBD

The functional block diagram is a standard type of graphical presentation.

The following basic symbols are available for creating functional block diagrams.



AND–block

OR–block

Input
Negated input

The function block diagram operations are parameterized with operands/data structures (for data structures see also chapter 4.6).

OBs, PBs and FBs can be programmed in FBD.

The end of a block is marked by a box with ”Block end”.

For explanation of terms, see Bild 4.

Branching instructions are not permitted in functional block diagram, but block calls are.

Note: More detailed information, e.g. on generating programs in functional block diagram, is contained in the user instructions ”Configuration”.

Chapter 4
Software Blocks

In the following chapter, the most important functions and features of the various blocks are described.

4.1 Block types

- The **organization block OB**
  contains the overall structure and defines the order in which the other blocks are to be processed.

- The **program block PB**
  brings together parts of a user program, such as modules, machine parts and plant sections, from the technological point of view.

- The **function block FB**
  processes frequently recurring program parts as separate subroutines.

- The **standard function block SFB**
  has the same function as the FB and is an integral part of the standard PLC features.

Individual networks, represented in IL, LD or FBD, are the ”infrastructure” of which program blocks and function blocks (FBs and SFBs) are composed. The individual networks form the program with its sequence of instructions for the process control concerned.

The user program in Dolog AKF is composed of various blocks. The selection depends on the complexity of the task definition and the requirement to keep the configuration as simple as possible. Block technology thus allows structured programming.

- The **SYM/COM block**

  - **in A250**
    contains and organizes the assignment of hardware addresses, symbolic addresses and comments. It cannot be linked in a network and is instead generated separately with the SYM/COM editor.

  - in A120 with data blocks DB0....DB9
    contains and organizes the assignment of hardware addresses, symbolic addresses and comments. It cannot be linked in a network and is instead generated separately with the SYM/COM editor.

4.2 Organization blocks OB

In AKF125, organization blocks are available.

The organization blocks can be generated in IL, LD or FBD.

OBs are processed cyclically. Each scan (cycle) begins with the processing of network 001 and ends with the processing of the last network contained in the OB.

From within the OB, the program and function blocks PB, FB and SFB are called and processed in the required order.

PBs and FBs consist of a series of consecutively numbered networks beginning with network 001.

Each network contains only one PB-, FB-, AZ- or SFB-call (IL is an exception) or one user program part in IL, LD or FBD.

All further OBs are available for control loops and interrupt processing.

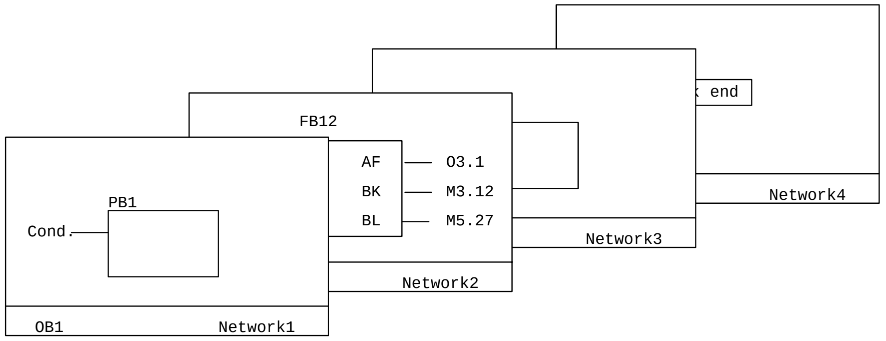


Figure 5 Example of networks in an organization block

A block call is followed by processing of the respective block (PB, FB or SFB), possibly condition–dependent. This is followed by a return to the next network in the OB.

4.3 Program block PB

A program block generally contains technologically related sections of the user program, e.g. one of x different machines.

Program blocks can be generated in IL, LD or FBD.

Structure:
A PB consists of a series of consecutively numbered networks, beginning with network 001. In these networks you can create IL, LD or FBD program parts, or you can call conditional or unconditional PBs, FBs and SFBs.

Call:
PBs are called from an OB, another PB or an FB.

You can call one and the same PB many times.
A block to be called is displayed in a square (in LD/FBD) in the network. The PB number is above the square. In the case of conditional PBs, the signal address of the condition is to the left of the square.
A PB that is not called from any place is never processed.

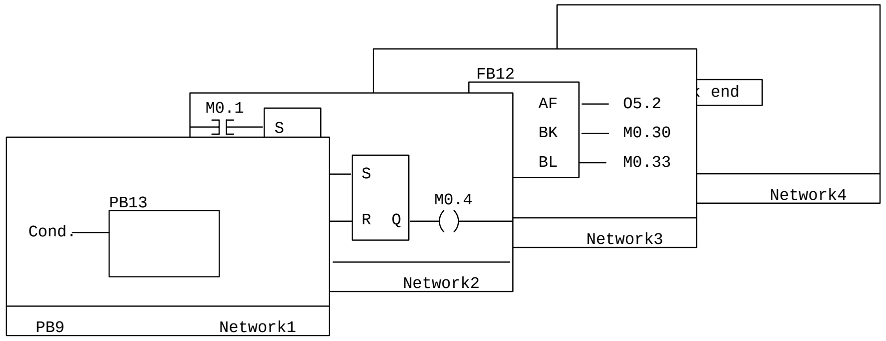


Figure 6 Example of networks in a program block

4.4 Function block FB

FBs are used to create frequently recurring program parts. They are parameterizable subroutines, i.e. an FB can be called and parameterized many times in various places.

Function blocks can be generated in IL, LD and FBD.

An FB can call PBs and FBs.

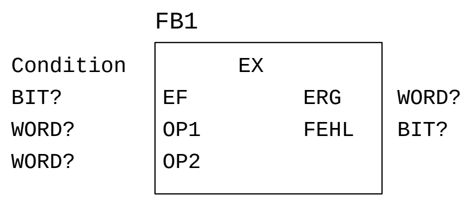
It is necessary to distinguish between a function block and a function block call. A function block contains part of a user program. A function block call ensures that the FB is processed during runtime, at precisely the point where the user program reaches that call. Before a block is processed, the parameter specifications of the FB are transferred to the subroutine (formal operands are replaced by actual operands/data structure). An FB that is not called from any place is also never processed.

Structure:
The program of a function block consists of a declaration part and an instruction part.

- **Declaration part**
  The declaration part is always located in the first network of an FB.
  In the declaration part, you enter the names of the function blocks and a list of the formal operands, and specify the type.
  The declaration part also includes information on the graphic structure of the block square and the order of the parameters.
  The declaration part can be altered later, within limits.

- **Instruction part**
  The instruction part contains the program in IL, LD or FBD. This program sets up the algorithmic correlations between the formal operands in the declaration part.
  The names of the formal operands in the instruction list must always be preceded by an ”=” sign.
  To the right of the instruction list, in appropriate cases, there is a number that states the nesting depth of the line concerned.

Call



Conditional call of a function block in ladder diagram or function block diagram. The formal operands are shown in the block.
The question mark indicates where actual operands/data structures (M3.3, A3.1, WLIN1.1) are required.

Figure 7 Example of a conditional call

An FB can be called from another FB, PB or OB. The FB is then displayed as a square in LD/FBD network.

Its name, in short form, is shown again in the square. The input formal operands appear on the left within the square, and the actual operands/data structures are outside the square and to the left, possibly with a condition above them. The output formal operands appear on the right within the square, with the actual operands/data structures outside the square and to the right. After the FB call, you simply enter the parameter outside the square.

If you change formal operands in the declaration part of an FB, you must reparameterize all affected FB calls. You can quickly find where the FB calls for the FB concerned are located in the user program, with the aid of the program overview or the global cross reference list.

You can also call an FB from the instruction part of another function block (nesting, recursion). The called function block can contain the formal operands of the calling FB, within its instructions, as actual operands.

The FB is limited in code length and can contain approximately 2,000 instructions. The number of instructions is reduced by the number of characters used for comments, labels and parameters (1 character = 1 byte).

4.5 Standard function block SFB

This Dolog software was sent to you complete with a library of standard function blocks. These blocks are predefined and need only be called (conditionally or unconditionally) from a place chosen by the user, and parameterized.

The declaration part and instruction part of the SFB are already in the software and cannot be changed later by the user. The formal operands are specified. The configuration planner calls the block at the required point in the program and parameterizes it with actual operands or data structures as required (see also chapter 4.6).

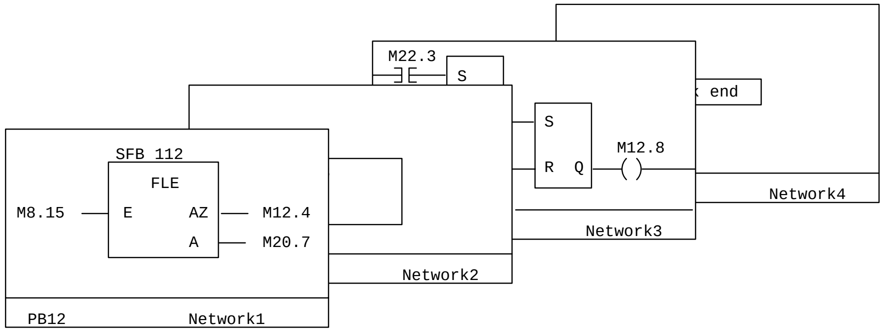


Figure 8 Example of an SFB call in the network of a PB (NW1)

4.6 Data structures for A250

The data structure contains a table of parameters that technologically belong together. A distinction is made between:

- AEG data structures
  - are strictly predefined, not modifiable by the user
  - are used for parameterizing SFBs or are required internally by AKF125 (e.g. setpoint/actual value fields of the intelligent function blocks)

- User’s own data structures
  - are generated completely, including attributes, parameter lists, etc., by the user
  - are used for parameterizing user function blocks;individual parameters (called data structure elements) can also be linked to SFBs

All data structures can be seen in AKF125 menu ”Edit”, ”Data structures”.

Stipulations for AEG data structure elements are made in the menu ”Edit”, ”Symbols and comments”.

User data structures are created and defined in the menu ”Edit”, ”Data structures”. Stipulations for the elements are made in the Menue ”Edit”, ”Symbols and comments”.

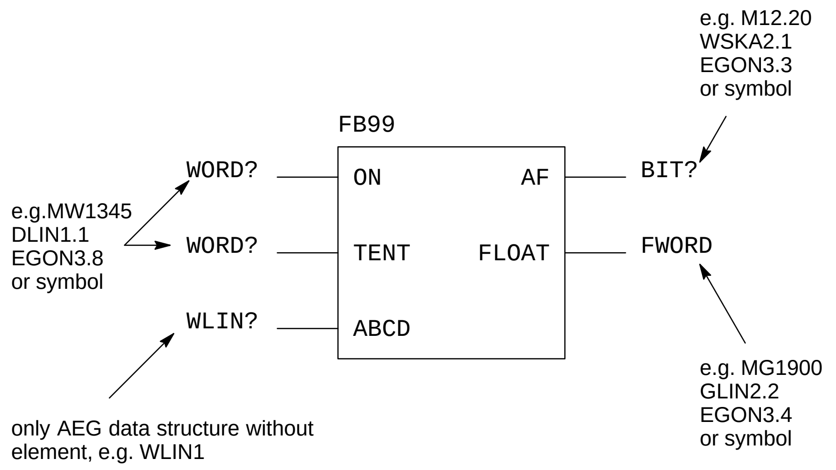
The following table and sample FB illustrate the distinction between the various groups.

Table ¿+ 2 Definition characteristics of data structures

Ctiteria	**AEG data structure**	**User data structures**	
		AKF125/ALD25	**Import from ASCII**
Generator	AEG	User	User
in signal memory (near=yes far=no)	yes, no	no	no
Data structure editor	visible	generatable	visible
Names e.g.	1...6 letters GSKA, WLIN	4 or more letters HUGO, EXAMP	4 or more letters HUGO, EXAMP
Element types	BIT, BYTE, WORD, DOUBLE WORD, FLOATING WORD, POINTERFLOATING STREAM	BIT, BYTE, WORD, DOUBLE WORD, FLOATING WORD, POINTER, WORD,	BIT, BYTE, WORD, DOUBLE WORD, STREAM
Attributes, user as– signable	none	read,write, syswrite, element type (see above), Display in SYM/COM, Initial values, on–line exchangeable	read,write, syswrite, element type (see above), Display in SYM/COM, Initial values, on–line exchangeable, exists as external file
Usable in FB as formal operand	U GSKA1.3 corresponds to U =OPA	U HUGO5.7 corresponds to U =OMA	U EXAMP3.9 corresponds to U =EVA

☞ **Note:** A list of the AEG data structures in SYM/COM blocks can be found in the ”A250 Pocket Guide” or ”A250 Block Library”.

The following FB contains all data structure types, as an example. The assignment of formal operands and data structures takes place in the declaration part. In this case, only the FB call is given.


e.g. M12.20
WSKA2.1
EGON3.3
or symbol
FB99
WORD?
ON
AF
BIT?
e.g.MW1345
DLIN1.1
EGON3.8
or symbol
WORD?
TENT
FLOAT
FWORD
WLIN?
ABCD
e.g. MG1900
GLIN2.2
EGON3.4
or symbol
only AEG data structure without
element, e.g. WLIN1

4.7 SYM/COM block for A250/ Data block DB0...DB9 for A120

To make the relationship between an absolute address (I/Os, markers, etc.) and its technological function clear, it is possible to provide absolute addresses with symbolic names and comments.
The symbolic names and comments text is stored in the SYM/COM block under the actual station name.
After activating the SYM/COM block, the alternatives to the absolute addresses, i.e. the symbolic names entered in the SYM/COM block, can be used in programming.
The SYM/COM block can be also be documented.

Signal	Symbol	Comment	Initial value *)
I3.1	ON	Motor 1 on	
I3.2	MOT_RI	Motor right on	
I3.3	MOT_LE	Motor left on	
I3.4	STOP	Esc key	
I3.5	PUMP_1	Pump 1 on	
I3.6	PUMP_2	Pump 2 on	
I3.7			
I3.8			
I3.9			
I3.10			
I3.11	TONG_UP	Tongs up	
I3.12	TONG_DO	Tongs down	
I3.13			
I3.14			
I3.15			
I3.16			

*) not for A120

Figure 9 Examples of SYM/COM block entries

Chapter 5
First programming steps A120/AKF125 with ALU 200, 201, 202

This chapter includes an example of a small application, complete in all details, for programming the A120 with AKF125.

5.1 Introduction

This chapter will show you, as a beginner in AKF, how to take the first steps in programming an A120. For this, a simple program will be created in AKF125, loaded into the PLC and then viewed using dynamic status display.

5.2 Preparations

The following preparations should already have been made:

- ❐ On the programming unit (in this case a P610C) you will find the software installed on drive C: (see ”Installation” in the user instructions).

☞ **Note:** The sample program is installed along with the software.

- ❐ Included in the example is an A120 with the following modules:
  Subracks DTA 112, ALU 200, DEP 216 (on slot reference 2), DAP 216 and two SIM 011s as simulators on DEP inputs I2.1 to I2.16
  Please also remember the PaDT connection cable ↔ PLC.

5.3 Task definition

The idea is to create a program where an 8 bit wide bit string traverses on 16 bits of an output module (”running light”). The bit string which is going to be used is set by using inputs I2.9 to I2.16 and accepted via input I2.2. It is possible to stop the output by using I2.1 = 1 (all 16 outputs = 0) or to ”freeze” the current status by using I2.3 = 0. Programming is carried out in the special language Instruction List and symbolically.

The sample plant is called ”EXERCISE”, the sample program is called ”RUN12”.

☞ **Note:** The program logic already exists; what we have here is an exercise in handling AKF

5.4 Wiring diagram

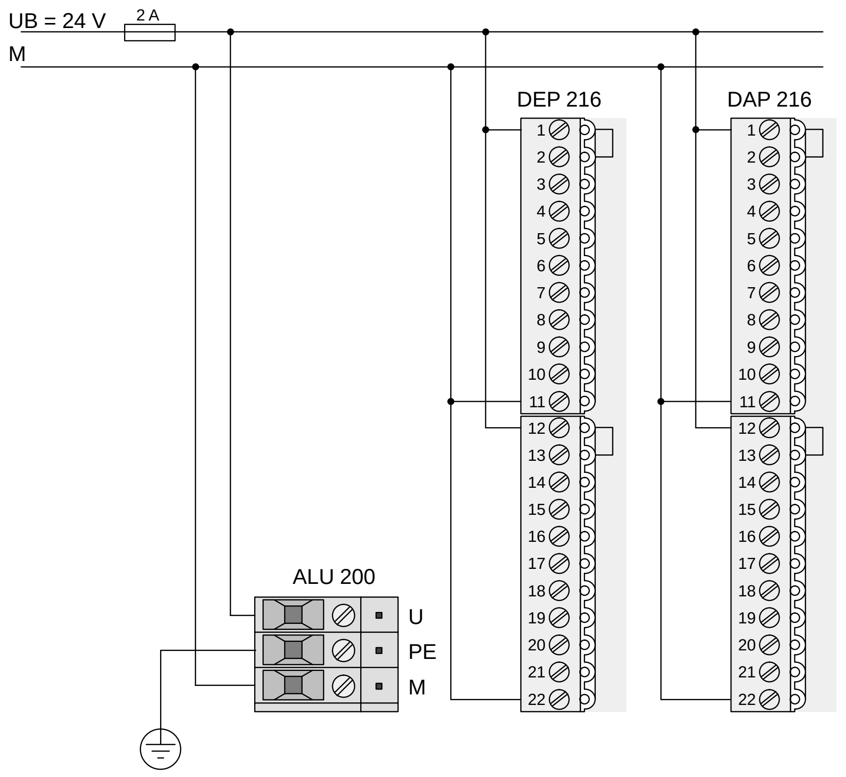


Figure 10 Hardware power supply

5.5 Parameters of the sample program

The following parameters are used in the program:

Table 1 Operands in the sample program

Operand	Symbol	Comment
I2.1	OFF	Off=1: all outputs off, Off=0: display
I2.2	LOAD	0->1 edge loads bit string
I2.3	FREE	Free=0: freeze, Free=1: run
I2.9	BIT1	First bit of bit string
I2.10	BIT2	Second bit of bit string
I2.11	BIT3	Third bit of bit string
I2.12	BIT4	Fourth bit of bit string
I2.13	BIT5	Fifth bit of bit string
I2.14	BIT6	Sixth bit of bit string
I2.15	BIT7	Seventh bit of bit string
I2.16	BIT8	Eighth bit of bit string
Q3.1	RUN1	
Q3.2	RUN2	
Q3.3	RUN3	
Q3.4	RUN4	
Q3.5	RUN5	
Q3.6	RUN6	
Q3.7	RUN7	
Q3.8	RUN8	Outputs on which the bit string is alternately sent (running light)
Q3.9	RUN9	
Q3.10	RUN10	
Q3.11	RUN11	
Q3.12	RUN12	
Q3.13	RUN13	
Q3.14	RUN14	
Q3.15	RUN15	
Q3.16	RUN16	
M15.1	HELP1	Help marker for edge detection
M15.2	HELP2	Help marker for edge detection
M15.3	HELP3	Help marker for edge detection
M15.4	HELP4	Help marker for edge detection
MW72	ROTATED	This word contains the rotated info
SM15	PULSE	10.0 Hz flashing rate

5.6 Programming

☞ **Note:** Menu functions are declared in ”inverted commas”, e.g. ”Edit”, ”Blocks”. Entries that you input (type) are in `Courier`, e.g. `AKF12`. Key combinations/special keys are shown in brackets, e.g. <Ctrl>+<S>.

5.6.1 Call program

Step 1	Call the software from user drive C: with the following command: `AKF125`

☞ **Note:** The following steps apply for the first software call after installation. If the AKF125 software has already been used for configuration and a plant has already been created for A250, ignore the following steps and refer to the chapter ”Migration from A250 to A120 configuration”.

Response	You will be asked to enter an A250 plant name.
Step 2	Enter a plant name of your choice, e.g.: `C:\PLANT`, then press <Return>
Response	You will be asked whether you wish the plant to be created.
Step 3	Confirm this by pressing Y for Yes or move the menu bar to YES with the arrow keys and press <Return>.
Response	You will be asked to enter an A250 station name.

Step 4 Enter the station name of your choice, e.g.: `STATION`, and press <Return>

Response You will be asked whether you wish the station to be created.

Step 5 Confirm by pressing Y for Yes or move the frame to YES with the arrow keys and press <Return>.

Response The main menu for A250 will be displayed.

5.6.2 Migration from A250 to A120 configuration (via ALU Group)

☞ **Note:** The marked capitals displayed in a different color (reference characters) are for calling the menu directly.
All the steps mentioned below should be carried out in the order shown (even when the numbering begins again at ”1” in every sub–step).

Step 1 Enter T for ”SeTup”

Response The setup menu opens

Step 2 Enter S for ”PLC station”

Step 3 Enter A for ”ALU Group” and keep toggling (= press reference character or <Return>) until ”200; 201; 202” appears.

Step 4 Press <Esc> to exit the Station menu

Response You are asked whether you really want to select this ALU Group.

Step 5 Confirm by pressing Y for Yes or move the frame to YES with the arrow keys and press <Return>.
From here on we shall just refer to this as ”Confirm with Yes”.

5.6.3 Creating a plant/station

☞ **Note:** If the AKF125 software has already been used for configuration and a plant has already been created for A120 erzeugt worden, you will need to make the following entries before steps 1-4:
1) Enter T for ”SeTup”
2) Enter A for ”PlAnt”
However, note from the next chapter onwards that you still in the setup menu.

Response You will be asked to enter an A120 plant name.

Step 1 Enter the following plant name:
`C:\EXERCISE`, then press <Return>

Response You will be asked whether you wish the plant to be created.

Step 2 Confirm with Yes

Response You are asked to enter an A120 station name.

Step 3 Enter the following station name:
`RUN12`, then press <Return>

Response You will be asked whether you wish the station to be created.

Step 4 Confirm with Yes

Response The main menu for A120 will be displayed.

5.6.4 Programming presettings

We will now take a look at the setups.

Step 1 Enter `S` for ”PLC station”

Response The presettings menu opens.

If the presettings are not yet as you want them, carry out the following settings:

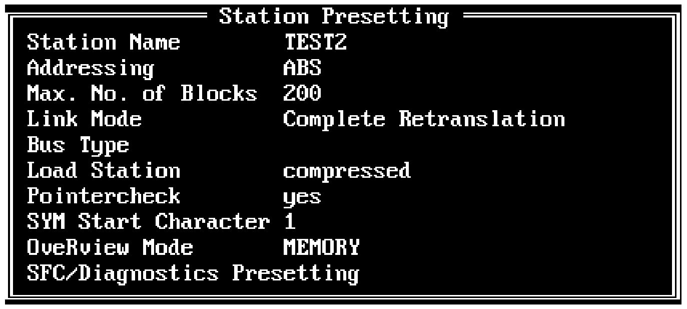
Step 2 Enter `A` for ”Addressing”; keep toggling until ”SYM” appears (symbolic programming)

Step 3 Enter `T` for ”Data block number”, press <Return> and type `0` . Press <Return> again to finish

Step 4 Enter `B` for ”Max. No. of Blocks”. Enter `10` <Return>

Step 5 Enter `M` for ”Link Mode”, select ”Complete Retranslation” with the arrow keys and press <Return> again

Step 6 Enter `V` for ”Memory Variant” and keep toggling until ”RAM” appears



```
Station Presetting
Station Name        TEST2
Addressing          ABS
Max. No. of Blocks  200
Link Mode           Complete Retranslation
Bus Type
Load Station        compressed
Pointercheck        yes
SYM Start Character 1
OveRview Mode       MEMORY
SFC/Diagnostics Presetting
```

Step 7	Press <Esc> twice
Response	The menus close and the cursor bar stays on ”SeTup” in the main menu line. The setups have now been accepted.

5.6.5 Equipment list

Step 1	Interconnect the PaDT with the PLC (via connection cable YDL 52)
Step 2	Switch on the PLC power supply
Step 3	Enter L for ”Load”
Response	The load menu opens
Step 4	Enter L for ”Read Equipment List”
Response	The equipment list is read from the PLC and you are asked whether the existing equipment list is to be overwritten.
Step 5	Confirm with Yes

Let us now take a look at the equipment list:

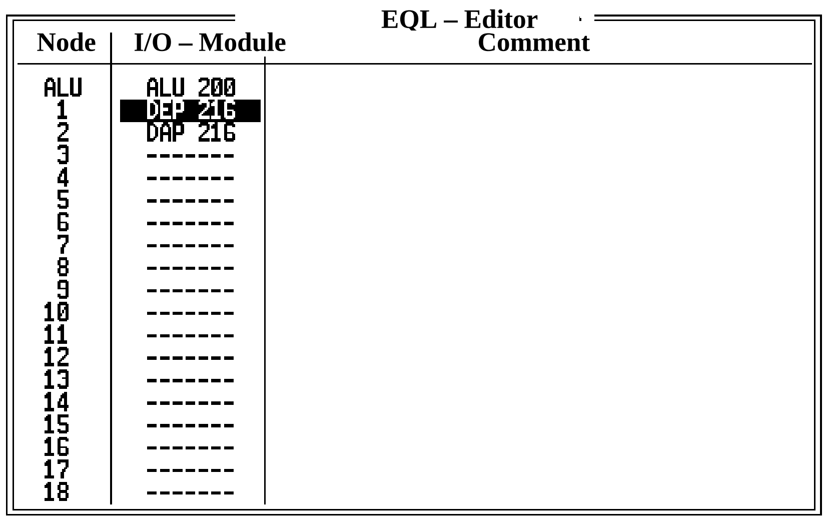
Step 1 Use <←> to go to ”Edit”

Response The edit menu opens.

Step 2 Enter L for ”Equipment List”

Response The equipment list editor opens and the component selection that has been read out can be viewed.

The following figure shows how your equipment list should now look.



```
                         EQL – Editor
  Node | I/O – Module |            Comment
  ALU  |  ALU 200     |
   1   |  DEP 216     |
   2   |  DAP 216     |
   3   |  -------     |
   4   |  -------     |
   5   |  -------     |
   6   |  -------     |
   7   |  -------     |
   8   |  -------     |
   9   |  -------     |
  10   |  -------     |
  11   |  -------     |
  12   |  -------     |
  13   |  -------     |
  14   |  -------     |
  15   |  -------     |
  16   |  -------     |
  17   |  -------     |
  18   |  -------     |
```

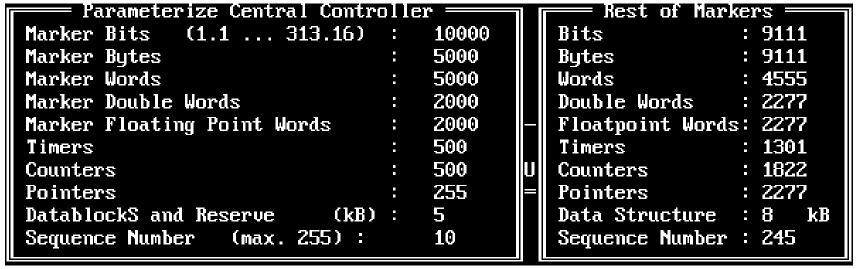
5.6.5.1 Edit menu for the equipment list editor

This menu is used mainly for configuring the secondary subrack (mounted components, addresses, timeout, etc.). You can also use it to define the segmentation of the signal memory for the controller. In the following example look at the presetting only.

Step 1 Enter <Ctrl>+<Return> to open the menu

Step 2 Enter Z for ”ParameteriZe Central Controller”

Response The following menu appears (presettings)



```
═══ Parameterize Central Controller ═══        ═══ Rest of Markers ═══
Marker Bits   (1.1 ... 313.16)  :   10000      Bits            : 9111
Marker Bytes                    :   5000       Bytes           : 9111
Marker Words                    :   5000       Words           : 4555
Marker Double Words             :   2000       Double Words    : 2277
Marker Floating Point Words     :   2000    ─  Floatpoint Words: 2277
Timers                          :   500        Timers          : 1301
Counters                        :   500     U  Counters        : 1822
Pointers                        :   255     =  Pointers        : 2277
DatablockS and Reserve     (kB) :   5          Data Structure  : 8   kB
Sequence Number   (max. 255) :      10         Sequence Number : 245
```

Step 3 Press <Esc> to exit the menu

5.6.5.2 Terminate equipment list and save

Step 1 Press <Ctrl>+<Return> to open the menu

Step 2 Press T (for Terminate); the equipment list is saved and you exit the equipment list editor.

Response The edit menu can be viewed again.

5.6.6 Assign symbols and comments (SYMCOM block)

As symbolic programming is intended, the signal symbols have to be defined. Ideally, this should happen before the program is created.

☞ **Note:** This is spread over several chapters.

Step 1 Enter `C` for ”Symbols and Comments”

Response The menu for ”Symbols and Comments” opens.

Step 2 Enter `S` for ”Start Entry”

Response The Symbols and Comments editor opens. All I/Os that are unavailable according to the equipment list are highlighted with a * sign. The ”Symbol” column is written in upper case and the ”Comment” column in lower case.

5.6.6.1 Assign symbols for inputs

Step 3 Press <Ctrl>+<Return> to open the processing menu

Step 4 Enter `S` for ”Search Function”

Response A window opens in which you can enter the signal you are looking for.

Step 5 Enter `I2.1` and <Return>

Response The cursor jumps to the ”Symbol” column for signal I2.1.

Step 6	Enter `OFF` and <Tab>
Response	The cursor jumps to the ”Comment” column
Step 7	Enter as comment: `Off=1: all outputs off, Off=0: display` and confirm with <Tab>
Response	The entry is made in line I2.1 and the cursor is now on the line for signal I2.2.
Step 8	Enter: `LOAD` <Tab> `Load the bit string by using edge 0–>1` <Tab>
Response	The entry is made in line I2.2 and the cursor is now on the line for signal I2.3.
Step 9	Enter: `FREE` <Tab> `Free=0: freeze, Free=1: run` <Tab>
Response	The entry is made in line I2.3 and the cursor is now on the line for signal I2.4.
Step 10	Enter <Ctrl>+<Return> to open the processing menu
Step 11	Enter `S` for ”Search Function”
Step 12	Enter `I2.9` and <Return>
Response	In the editor the cursor jumps to the ”Symbol” column of the specified signal.

Step 13 Enter the following text for I2.9 to I2.16:

```
BIT1 <Tab> First bit of bit string <Tab>
BIT2 <Tab> Second bit of bit string <Tab>
BIT3 <Tab> Third bit of bit string <Tab>
BIT4 <Tab> Fourth bit of bit string <Tab>
BIT5 <Tab> Fifth bit of bit string <Tab>
BIT6 <Tab> Sixth bit of bit string <Tab>
BIT7 <Tab> Seventh bit of bit string <Tab>
BIT8 <Tab> Eighth bit of bit string <Tab>
```

Response The entries were made in lines I2.9 to I2.16 and the cursor is now on the line for the next signal: *I3.1.

Step 14 Enter <Ctrl>+<Return> to open the processing menu

Step 15 Enter S for ”Search Function”

Step 16 Enter Q3.1 and <Return>

Response In the editor the cursor jumps to the ”Symbol” column of the specified signal.

5.6.6.2 Assign symbols for outputs

Step 17 Enter the following text for Q3.1 to Q3.16:

```
RUN1 <Tab> Outputs Q3.1-16 are <Tab>
RUN2 <Tab> used for the running light.<Tab>
RUN3 <Tab> The bit string set at <Tab>
RUN4 <Tab> I2.9-16 traverses if <Tab>
RUN5 <Tab> a 0->1 edge  <Tab>
RUN6 <Tab> is given. <Tab>
RUN7 <Tab> <Tab>
RUN8 <Tab> <Tab>
RUN9 <Tab> <Tab>
RUN10 <Tab> <Tab>
RUN11 <Tab> <Tab>
RUN12 <Tab> <Tab>
RUN13 <Tab> <Tab>
RUN14 <Tab> <Tab>
RUN15 <Tab> <Tab>
RUN16
```

5.6.6.3 Assign Symbols / Comments for marker / marker word

Step 18 Enter <Ctrl>+<Return> to open the processing menu

Step 19 Enter `S` for ”Search Function”

Step 20 Enter `MW72` and <Return>

Response The cursor jumps to the first symbolic character of signal MW72.

Step 21 Enter at MW72:
`ROTATED` <Tab> `This word contains the rotated info`

Response When the o of info is entered the line is full, a warning tone is heard and the cursor jumps automatically to the next line.

Step 22 Enter <Ctrl>+<Return> to open the processing menu

Step 23 Enter `S` for ”Search Function”

Step 24 Enter `M15.1` and press <Return>

Step 25 Enter the following text for M15.1 to M15.4:
`HELP1` <Tab> `Help marker for edge detection` <Tab>
`HELP2` <Tab> `Help marker for edge detection` <Tab>
`HELP3` <Tab> `Help marker for edge detection` <Tab>
`HELP4` <Tab> `Help marker for edge detection` <Tab>

Step 26 Enter <Ctrl>+<Return> to open the processing menu

Step 27 Enter `S` for ”Search Function”

Step 28 Enter `SM15` and <Return>

Step 29 Enter the following text: `PULSE`

Step 30	Enter <Ctrl>+<Return> to open the processing menu
Step 31	Enter T for ”NW Terminate”
Response	This finishes with editing the SYM/COM block and the texts are saved.
Step 32	Enter <Esc>
Response	The edit menu can be viewed again.

5.6.7 Edit program (blocks)

This chapter deals with entering the user program in AKF12.

5.6.7.1 Open block editor

Step 1	Enter `B` for ”Blocks”
Response	The block menu opens.
Step 2	Enter `B` for ”Block number”
Step 3	Enter `1` and press <Return>
Step 4	Enter `N` for ”Network number”
Step 5	Enter `1` and press <Return>
Step 6	Enter `T` for ”Block Type” and keep toggling until ”PB” appears
Step 7	Enter `M` for ”Input Mode” and keep toggling until ”IL” appears
Step 8	Enter `A` for ”Addressing”: keep toggling until ”SYM” appears
Step 9	`S` for ”Start Entry”
Response	The block editor opens and the last network of the PB1 appears with ”BE” for block end.

5.6.7.2 Edit PB1

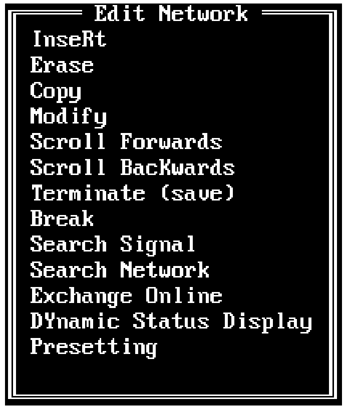
First the program block (PB) with the running light program is edited. The PB1 consists of network 1 with the program and network 2 with ”BE” for block end.

❐ **Insert network**

A blank block only ever consists of network 1, which is where the block end is located. In order to edit within the block, you must first insert a blank network.

Step 1 Press <Ctrl>+<Return>

Response The processing menu opens



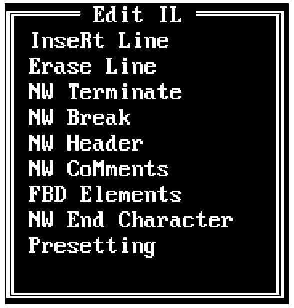
Step 2 Enter R for ”InseRt” (Network)

Response With the ”Insert” function, a new network is always inserted in front of the current network. In this case network 1 is now blank (contains only :***); network 2 shows ”BE” for block end.

❐ **Edit network 1**
In network 1, the instructions for the program are now created. The program contains some jumps, which are generally edited in instruction list. Therefore the entire program will be created in instruction list, though this is not absolutely necessary.

Step 1 Press <Ctrl>+<Return>

Response The processing menu opens



```
Edit IL
InseRt Line
Erase Line
NW Terminate
NW Break
NW Header
NW CoMments
FBD Elements
NW End Character
Presetting
```

Module

Step 2 Press R for ”InseRt Line”

Step 3 Press <Ctrl>+<E> several times in order to add more lines (the program has 26 lines in total)

Edit Module

```
:
:
:
:
:
:
:
:
:***
```

Step 4 Now enter the following lines. In the process, you can add more blank lines above the cursor during editing by pressing <Ctrl>+<E>.

`A` <Tab> `OFF` <Return>
`JF` <Tab> `=JUMP1` <Return>
`L` <Tab> `K0` <Return>
`=` <Tab> `ROTATED` <Return>
`L` <Tab> `K0` <Return>
`JI` <Tab> `=JUMP3 <Return>`

Step 5 For jump target JUMP1, use the arrow key to move the cursor left along the blank line as far as the edge of the screen:
`JUMP1` <Tab> (to :) `A` <Tab> `LOAD` <Return>

Step 6 Enter the following lines in accordance with Schritt 4:
`EDP` <Tab> `HELP1` <Return>
`=C` <Tab> `HELP2` <Return>
`JF` <Tab> `=JUMP2` <Return>
`LBW` <Tab> `BIT1` <Return>
`DBB` <Tab> `CNT8` <Return>
`=` <Tab> `ROTATED` <Return>

Step 7 Enter the following line in accordance with Schritt 5:
`JUMP2` <Tab> (to :) `A` <Tab> `PULSE` <Return>

Step 8 Enter the following lines in accordance with Schritt 4:
`A` <Tab> `FREE` <Return>
`EDP` <Tab> `HELP3` <Return>
`=C` <Tab> `HELP4` <Return>
`JF` <Tab> `=JUMP4` <Return>
`A` <Tab> `ROTATED` <Return>
`ROL` <Tab> `V1` <Return>
`=` <Tab> `ROTATED` <Return>

Step 9 Enter the following line in accordance with Schritt 5:
`JUMP3` <Tab> (to :) `L` <Tab> `ROTATED` <Return>

Step 10 Enter the following lines in accordance with Schritt 4:
`TBW` <Tab> `RUN1` <Return>
`DBB` <Tab> `CNT16` <Return>

Step 11 Enter the following line in accordance with Schritt 5:
`JUMP4` <Tab> (to :) `NOP` <Return>
:***

- **Deleting superfluous blank instruction lines**

:*** completes the network. Redundant lines can be cleared as follows:

Step 1 Move the cursor to a blank line with the arrow keys

Step 2 Press <Ctrl>+<Return>

Step 3 Enter `L` for ”DeLete line”

- **Close network and save**

Step 1 Press <Ctrl>+<Return> to open the processing menu

Step 2 Enter `T` for ”NW Terminate”

- **Close block and save**

Step 1 Press <Ctrl>+<Return> to open the processing menu

Step 2 Enter `T` for ”Terminate (Save)”

Block PB1 is now complete and the ”Edit Blocks” menu is automatically displayed again. The PB1 must now be linked to the organization block, since this is where the ”threads” of the program are ”woven together”. Without the organization block, the program is not runnable.

5.6.7.3 Edit OB1

❒ **Open OB**

Step 1	Enter T for ”Block Type” and keep toggling until ”OB” appears
Step 1	Enter B for ”Block number”
Step 2	Enter 1 and press <Return>
Step 3	Enter S for ”Start Entry”
Response	The block editor opens and the last network of the OB1 appears with ”BE” for block end.
Step 4	Press <Ctrl>+<Return> to open the processing menu
Step 5	Enter R for ”InseRt” (Network)
Response	With the ”Insert” function, a new network is always inserted in front of the current network. In this case network 1 is now blank (contains only :***); network 2 shows ”BE” for block end.

❒ **Call PB in OB (unconditionally, i.e., the PB is called in every scan)**

Step 1	Press <Ctrl>+<Return> to open the processing menu
Step 2	Enter L for ”Insert Line”
Step 3	Enter the following text for the block call:
	BC <Tab> PB1 <Return>

❒ **Close the OB and save**

Step 1 Press <Ctrl>+<Return> to open the processing menu

Step 2 Enter T for ”Terminate” (network)

Step 3 Press <Ctrl>+<Return> to open the processing menu

Step 4 Enter T for ”Terminate” (block)

With this the program input is complete.

The following shows a printout of the program which has been created.

```
C:\UEBUNG\LAUF12\OB1
AEG Modicon Dolog AKF: Programm-Protokoll

NETZWERK: 0001

      :BA   PB 1
      :***

NETZWERK: 0002

      :BE

C:\UEBUNG\LAUF12\PB1
AEG Modicon Dolog AKF: Programm-Protokoll

NETZWERK: 0001

      :U    AUS
      :SPZ  =JUMP1
      :L    K 0
      :=    ROTIERT
      :L    K 0
      :SP   =JUMP3
JUMP1 :U    LADEN
      :FLP  HILF1
      :=C   HILF2
      :SPZ  =JUMP2
      :LBW  BIT1
      :DBB  ANZ 8
      :=    ROTIERT
JUMP2 :U    TAKT
      :U    FREI
      :FLP  HILF3
      :=C   HILF4
      :SPZ  =JUMP4
      :U    ROTIERT
      :ROL  K 1
      :=    ROTIERT
JUMP3 :L    ROTIERT
      :TBW  LAUF1
      :DBB  ANZ 16
JUMP4 :NOP
      :***
NETZWERK: 0002

      :BE
```

5.7 Link Program

The following function will tailor the program to PLC requirements

Step 1	To go to the main menu line, exit the open menus by pressing the <Esc> key.
Step 2	Enter L for ”Load”
Response	The load menu opens
Step 3	Enter L for ”Link Program”
Response	The program is linked
Step 4	Confirm the message

The program is now ready to be transferred to the PLC.

5.8 Transfer program to PLC

Using the following function, the program, the equipment list and the initial values are transferred to the PLC.

Step 1	Enter P for ”Program to PLC”
Response	The program has been transferred. It can now be started.

5.9 Start program

Step 1	Use <→> to switch to ”Online”
Response	The load menu closes and the online menu opens
Step 2	Press R for ”StaRt PLC”
Response	A message appears asking if you really want to start
Step 3	Confirm with Yes
Response	The yellow LED lights up on the ALU.

5.10 Setting and changing parameters

To obtain an output on Q3.1 to Q3.16, follow the steps described below (both si-mulators SIM 011 are attached to I2.1 to I2.16).

Step 1	Set up any bit string (0/1) on the switches on I2.9 to I2.16.
Step 2	Set I2.1 to ”1” and the ”0” (off)
Response	After initialization (I2.1 = 1) all outputs are off
Step 3	Set I2.3 to ”1”
Response	The running light is enabled (”run”)
Step 4	I2.2 from ”0” to ”1” (0 $\rightarrow$ 1 edge)
Response	The input bit string is accepted at the outputs and rotated. The program is now in its required end–state.

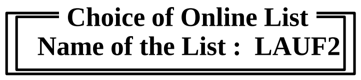
The bit string can be changed at any time and re–accepted by using I2.2.

5.11 Online List / Dyn. status display

5.11.1 Online List

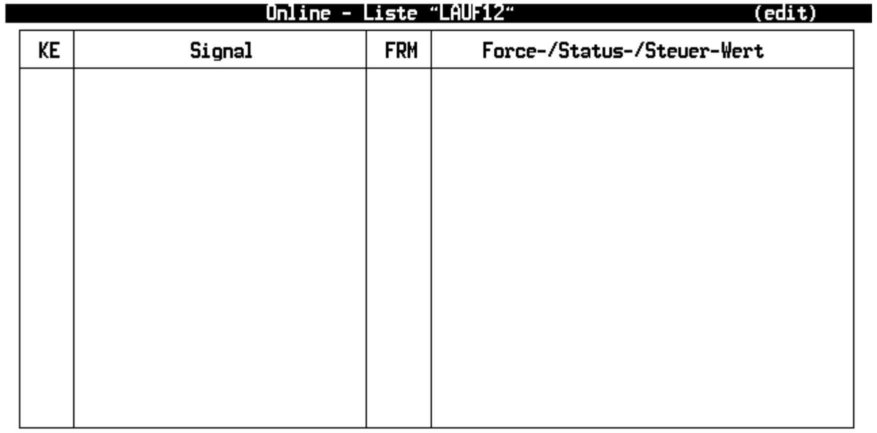
First a list is created in which you enter the signals. In the following example the list is used for displaying the status. Signals can also be controlled or defined (forced).

Step 1	After starting the PLC you will already be in the online menu. If not: press <Esc> to exit the open menus and then press `O` for ”Online”.
Step 2	Press `L` for ”Online List”
Response	The das Load/Erase On–line List menu appears
Step 3	Press `L` for ”Load Online List”
Step 4	Enter `RUN12` and press <Return>



Choice of Online List
Name of the List : LAUF2

Response	You will be asked whether you want to create a new list.
Step 5	Confirm with Yes
Response	The window with the list editor opens. You will find the cursor in the top, left–hand column.



Online - Liste "LAUF12" (edit)

KE	Signal	FRM	Force-/Status-/Steuer-Wert

Step 6 Enter the following text in the specified sequence:

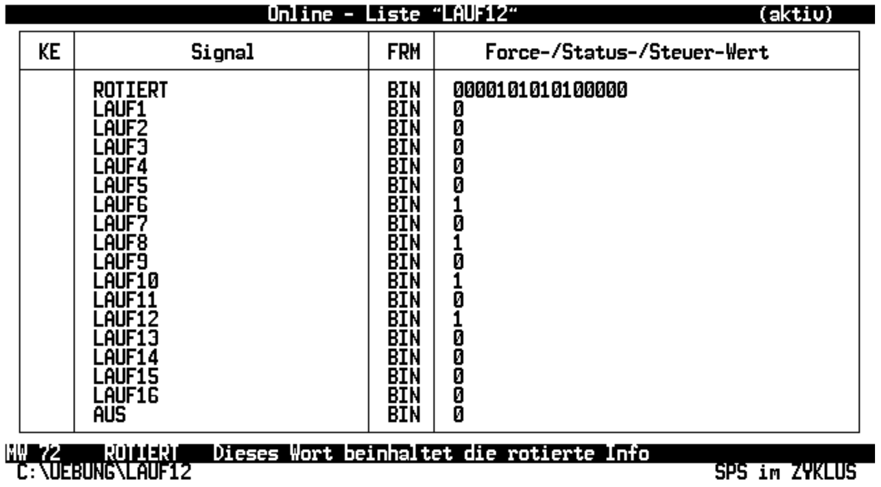
<Tab> `MW72` <Return> `B` <Return>
<Tab> `Q3.1` <Return>
<Tab> `Q3.2` <Return>
<Tab> `Q3.3` <Return>
.
.
.
<Tab> `Q3.16` <Return>
<Tab> `I2.1` <Return>
<Tab> `I2.2` <Return>
<Tab> `I2.3` <Return>
<Tab> `I2.9` <Return>
.
.
.
<Tab> `I2.16` <Return>

Response Signal input is now complete. You can now use the arrow keys or <PgUp>, <PgDn> to scroll through. (Ensure that the NUM-LOCK key is switched off.)

Step 7 Press <Ctrl>+<Return> to open the processing menu

Step 8 Enter U for ”StatUs display”

Response The top line switches to ”active”. To continue editing, switch back to ”edit” again by pressing <Esc>. The following figure shows an example of a status display using Online List.



```
                    Online - Liste "LAUF12"                   (aktiv)
 KE  |           Signal            | FRM |  Force-/Status-/Steuer-Wert
     |  ROTIERT                    | BIN | 0000101010100000
     |  LAUF1                      | BIN | 0
     |  LAUF2                      | BIN | 0
     |  LAUF3                      | BIN | 0
     |  LAUF4                      | BIN | 0
     |  LAUF5                      | BIN | 0
     |  LAUF6                      | BIN | 1
     |  LAUF7                      | BIN | 0
     |  LAUF8                      | BIN | 1
     |  LAUF9                      | BIN | 0
     |  LAUF10                     | BIN | 1
     |  LAUF11                     | BIN | 0
     |  LAUF12                     | BIN | 1
     |  LAUF13                     | BIN | 0
     |  LAUF14                     | BIN | 0
     |  LAUF15                     | BIN | 0
     |  LAUF16                     | BIN | 0
     |  AUS                        | BIN | 0
MW 72    ROTIERT   Dieses Wort beinhaltet die rotierte Info
 C:\UEBUNG\LAUF12                                          SPS im ZYKLUS
```

Step 9 Press <Ctrl>+<Return> to open the processing menu

Step 10 Enter T for ”NW Terminate”

Response You are asked whether you want to save the list.

Step 11 Confirm with Yes

Step 12 Exit the ”Online List” menu by pressing <Esc>

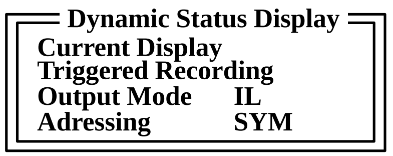
The online list is now complete. You can now also view signal status in the dynamic status display.

5.11.2 Dynamic Status Display

Now a run–through of the dynamic status display (”current display”).

Step 1 In the online menu: enter `D` for ”Dyn. Status Display”

Response The menu opens



```
Dynamic Status Display
Current Display
Triggered Recording
Output Mode     IL
Adressing       SYM
```

Step 2 Enter `L` for ”Current dispLay”

Step 3 Enter `B` for ”Block number”

Step 4 Enter `1` and press <Return>

Step 5 Enter `T` for ”Block Type” and keep toggling until ”PB” appears

Step 6 Enter `N` for ”Network number”

Step 7 Enter `1` and press <Return>

Step 8 Enter S for ”Start Display”

Response Network 1 of PB1 is selected

You can now scroll through the network using the arrow keys or scroll to the other network using <PgDn> and <PgUp>. You can press <Ctrl>+<Return> to open a processing menu in which various functions are available.

To view signal status scan by scan (e.g. for diagnostic purposes) use ”Triggered recording”.

5.12 Further practise (problem–solving)

A suggestion for further practise: modifying this program. Change the program so that the bit string is no longer set up with inputs, but with help markers.

- ❒ **Select the PB1 and network 1 under ”Edit”, ”Blocks”**
- ❒ **Select ”Modify” from the processing menu**
- ❒ **Change the bit string from I2.9 to I2.16 onto M12.1 to M12.8 by entering marker M12.1 instead of BIT1**
- ❒ **Close the network and the block**
- ❒ Transfer the resulting block online to the PLC **without** linking first (”Load”, ”Exchange Online”)
- ❒ **Then modify the online list by entering M12.1 ... M12.8 instead of Bit1 ... Bit8**
- ❒ **In the ”ID” column of the online list, enter ”CE” (for control enable) by the markers**
- ❒ **Then assign the markers the required bit string in the ”Force–/Sta-tus–/Control value” column**
- ❒ **Open the processing menu and enter the values in the PLC using ”Con-trol Enable”**
- ❒ Given 0 $\rightarrow$ 1 edge at I2.2, the new bit string is tranferred to the outputs.

5.13 Remarks about program documentation

You can use the ”Print” menu to carry out program documentation. If you use ”Entire Documentation”, all the important data are printed out in one run (complete with table of contents).

You can choose whether you want the lists to be output to screen, to a file or to the printer.

You can edit the files with any ASCII editor. You are in the current station directory. Assign the name of the file yourself when you select ”Output Unit”, ”File”.

When sending to the printer, make sure it has been initialized. Initialization is carried out in the ”SeTup”, ”Print” menu.

☞ **Note:** Please note the information on the documentation of menus in the ”Configuration” user instructions.

5.14 Remarks about data security

In the menu ”Special” you can archive the entire station (”Backup”) or restore.

☞ **Note:** Please note the information on the documentation of menus in the ”Configuration” user instructions.

Chapter 6
Initial programming moves AKF A250 / AKF125

This chapter includes an example of a small AKF application, complete in all details, for programming the A250 with the **AKF125** software package.

6.1 Introduction

This chapter will show you, as a beginner in AKF, how to take the first steps in programming an A250. For this, a simple program will be created in AKF125, loaded into the PLC and then viewed using dynamic status display.

6.2 Preparations

The following preparations should already have been made:

On the programming unit (in this case a P810) you will find the software installed on drive D: (see ”Installation” in the user instructions).

☞ **Note:** In this example, no Modnet 1/SFB installation has been carried out.

❒ Included in the example is an A250 with the following modules:
Subracks DTA 112, ALU 151, BIK 116 and DAP 102 (on slot reference 2) and two SIM 011s as simulators on DAP inputs I2.1 to I2.16
Please also remember the PaDT ↔ PLC connection cable.

6.3 Task definition

The idea is to create a program where an 8 bit wide bit string traverses on 16 bits of an output module (”running light”). The bit string which is going to be used is set by using inputs I2.9 to I2.16 and accepted via input I2.2. It is possible to stop the output by using I2.1 = 1 (all 16 outputs = 0) or to ”freeze” the current status by using I2.3 = 0. Programming is carried out in the special language Instruction List and symbolically.

The sample plant is called ”EXERCISE”, the sample program is called ”RUN25”.

☞ **Note:** The program logic already exists; what we have here is an exercise in handling AKF

6.4 Wiring diagram

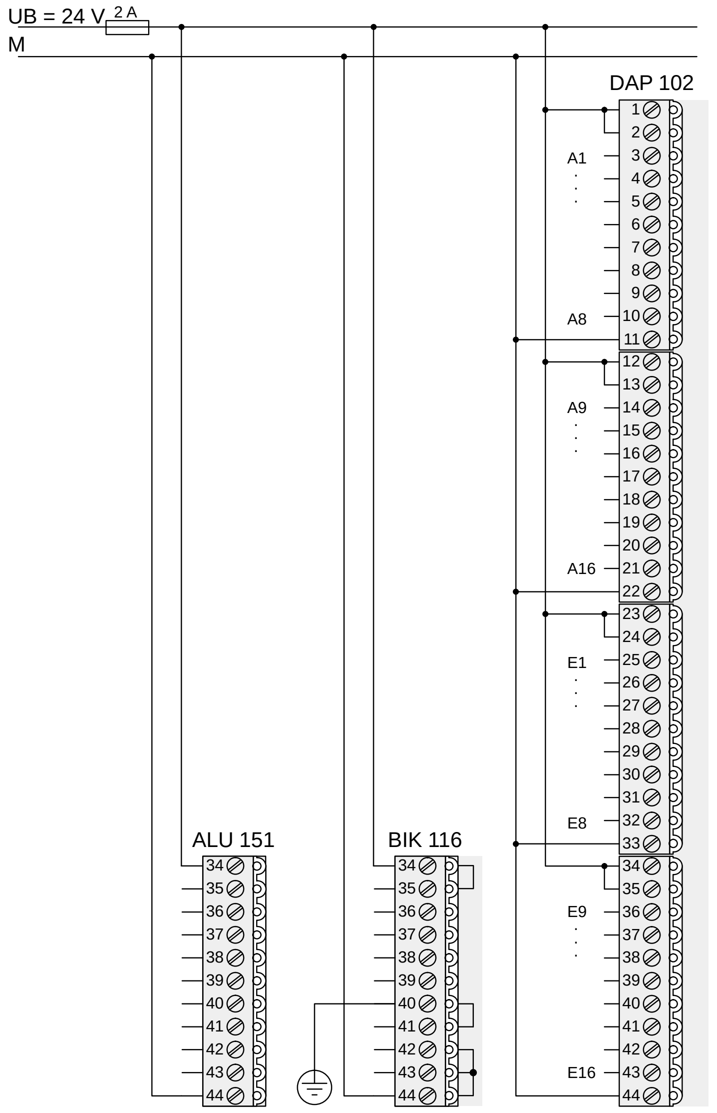


Figure 11 Hardware power supply

6.5 Parameters of the sample program

The following parameters are used in the program:

Table 2 Operands in the sample program

Operand	Symbol	Comment
I2.1	OFF	Off=1: all outputs Off=0: display
I2.2	LOAD	0->1 edge loads bit string
I2.3	FREE	Free=0: freeze, Free=1: run
I2.9	BIT1	First bit of bit string
I2.10	BIT2	Seconds bit of bit string
I2.11	BIT3	Third bit of bit string
I2.12	BIT4	Fourth bit of bit string
I2.13	BIT5	Fifth bit of bit string
I2.14	BIT6	Sixth bit of bit string
I2.15	BIT7	Seventh bit of bit string
I2.16	BIT8	Eighth bit of bit string
EW2.1	Bitstring	
A2.1	RUN1	Outputs on which the bit string is alternately sent (running light)
Q2.2	RUN2	
Q2.3	RUN3	
Q2.4	RUN4	
Q2.5	RUN5	
Q2.6	RUN6	
Q2.7	RUN7	
Q2.8	RUN8	
Q2.9	RUN9	
Q2.10	RUN10	
Q2.11	RUN11	
Q2.12	RUN12	
Q2.13	RUN13	
Q2.14	RUN14	
Q2.15	RUN15	
Q2.16	RUN16	
M15.1	HELP1	Help marker for edge detection
M15.2	HELP2	Help marker for edge detection
M15.3	HELP3	Help marker for edge detection
M15.4	HELP4	Help marker for edge detection
MW72	ROTATED	This word contains the rotated Info
SM173	TAKT_7	10.0 Hz flashing rate

6.6 Programming

☞ **Note:** Menu functions are declared in ”inverted commas”, e.g. ”Edit”, ”Blocks”. Entries that you input (type) are in `Courier`, e.g. `ALD25`. Key combinations/special keys are shown in brackets, e.g. <Ctrl>+<S>.

6.6.1 Call program

Step 1 Start the AKF125 software package from the DOS interface, e.g. from drive C:\ by entering AKF125 and then pressing Return.

☞ **Note:** The following steps apply for the first software call after installation. If the ALD25/AKF125 software has already been used for configuration and a plant has already been created, you will need to make the following entries before steps 2-5:
1) Enter T for ”SeTup”
2) Enter A for ”Plant”

Response You will be asked to enter an A250 plant name.

Step 2 Enter a plant name of your choice,
e.g.: `D:\EXERCISE`, then press <Return>

Response You will be asked whether you wish the plant to be created.

Step 3 Confirm this by pressing Y for Yes or move the menu bar to YES with the arrow keys and press <Return>.

Response You are asked to enter a station name.

Step 4 Enter the station name of your choice,
e.g.: `RUN25` , and press <Return>

Response You will be asked whether you wish the station to be created.

Step 5 Confirm by pressing Y for Yes or move the frame to YES with the arrow keys and press <Return>.

Response The main menu is displayed.

☞ **Note:** The marked capitals displayed in a different color (reference characters) are for calling the menu directly.
All the steps mentioned below should be carried out in the order shown (even when the numbering begins again at ”1” in every sub–step).

6.6.2 Migration from A120 to A250 configuration (via ALU Group)

☞ **Note:** The marked capitals displayed in a different color (reference characters) are for calling the menu directly.
All the steps mentioned below should be carried out in the order shown (even when the numbering begins again at ”1” in every sub–step).

Step 1 Enter T for ”SeTup”

Response The setup menu opens

Step 2 Enter S for ”PLC station”

Step 3 Enter A for ”ALU Group” and keep toggling (= press reference character or <Return>) until ”15x; 204; 205” appears.

Step 4	Press <Esc> to exit the Station menu
Response	Your are asked whether you really want to select this ALU Group.
Step 5	Confirm by pressing Y for Yes or move the frame to YES with the arrow keys and press <Return>. From here on we shall just refer to this as ”Confirm with Yes”.

6.6.3 Programming presettings

We will now take a look at the setups.

If the presettings are not yet as you want, carry out the following settings:

Step 1	Enter `T` for ”SeTup”.
Response	The setup menu opens.
Step 2	Enter `S` for ”PLC Station”.
Response	The Station menu opens.
Step 3	Call a new, predefined station by entering <Return>, <Blank>,<Return>.
Response	A selection window with the current AKF Stations opens.
Step 4	Select the required station with the cursor and accept by pressing <Return>.
Response	The new station is accepted into the Station menu.
Step 5	Enter `A` for ”Addressing”; keep toggling until ”SYM” appears (symbolic programming)
Step 6	`B` for ”Max. No. of Blocks”: enter `100` <Return>
Step 7	Answer the prompt with YES.
Step 8	`M` for ”Link Mode”: use the arrow keys to select ”Complete Representation” <Return>
Step 9	Select `T` for ”BusType”.
Response	Bus Type selection window opens.
Step 10	`S` for ”Modnet 1/SFB”: keep toggling until ”no” appears

Step 11 E for ”MMSE”: keep toggling until ”no” appears

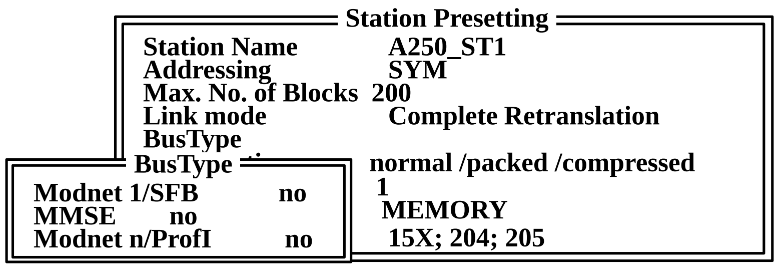
Step 12 I for ”ProfI: keep toggling until ”no” appears

Step 13 Exit the menu window with <ESC>

Step 14 Select L for Load Station. Keep toggling until the required load procedure overlays (**normal**, packed or compressed).

Step 15 Y for ”SYM–Start char.”: enter 1 <Return>

Step 16 R for ”OveRview mode”: keep toggling until ”MEMORY” appears



```
                 Station Presetting
  Station Name          A250_ST1
  Addressing            SYM
  Max. No. of Blocks  200
  Link mode             Complete Retranslation
  BusType
      BusType                normal /packed /compressed
Modnet 1/SFB        no        1
MMSE      no                  MEMORY
Modnet n/ProfI      no         15X; 204; 205
```

☞ **Note:** Menu item ALU Group retains the setting 15X ...

Step 17 Press <Esc> twice

Response The menus close and the cursor bar stays on ”SeTup” in the main menu line. The setups have now been accepted.

6.6.4 Edit equipment list

6.6.4.1 Activate equipment list editor

Step 1	Enter E for ”Edit”
Response	The edit menu opens
Step 2	Enter L for ”Equipment List”
Response	The equipment list editor is displayed. A menu offers a suggestion for the first subrack.

6.6.4.2 Set up subrack

Step 1	Confirm DTA 112 by pressing <Return>
Response	The first five lines are prepared for input (the grid fades out) and an ALU 151 is entered in the line for slot 0.
Step 2	If another ALU is preferred, press <Return>, selection ”ALU type”, and you can choose another ALU by pressing <Return>.

6.6.4.3 Enter modules

Step 1	Use <↓> to move the bar to the line for slot 1
Step 2	To open the module menu, press <Return>
Step 3	Enter S for ”Special”
Response	The special module menu opens
Step 4	Use <↓> to move the bar to BIK 116 and confirm with <Return>
Response	BIK 116 is entered under slot 1
Step 5	Use <↓> to move the bar to the line for slot 2

Step 6 To open the module menu, press <Return>

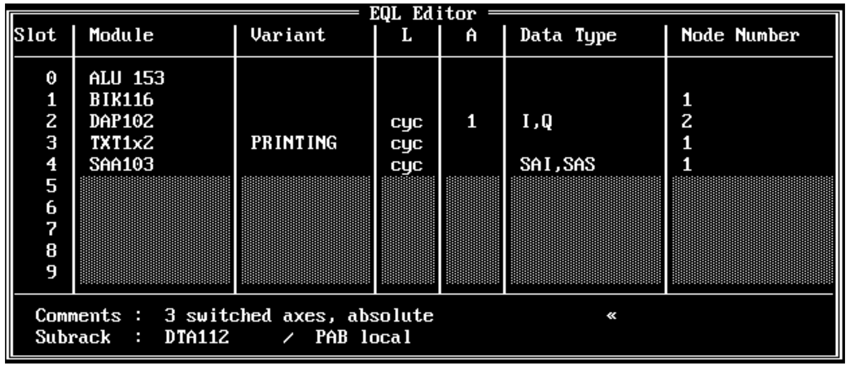
Step 7 Enter `D` for ”Digital I/O”

Response The digital module menu opens

Step 8 Use <↓> to move the bar to DAP 102 and confirm with <Return>

Response DAP 102 is entered under slot 2

The following figure shows how your equipment list should now look.



```
                              EQL Editor
Slot | Module      | Variant    | L   | A | Data Type  | Node Number

  0  | ALU 153     |            |     |   |            |
  1  | BIK116      |            |     |   |            | 1
  2  | DAP102      |            | cyc | 1 | I,Q        | 2
  3  | TXT1x2      | PRINTING   | cyc |   |            | 1
  4  | SAA103      |            | cyc |   | SAI,SAS    | 1
  5  |             |            |     |   |            |
  6  |             |            |     |   |            |
  7  |             |            |     |   |            |
  8  |             |            |     |   |            |
  9  |             |            |     |   |            |

  Comments :  3 switched axes, absolute                «
  Subrack  :  DTA112     /  PAB local
```

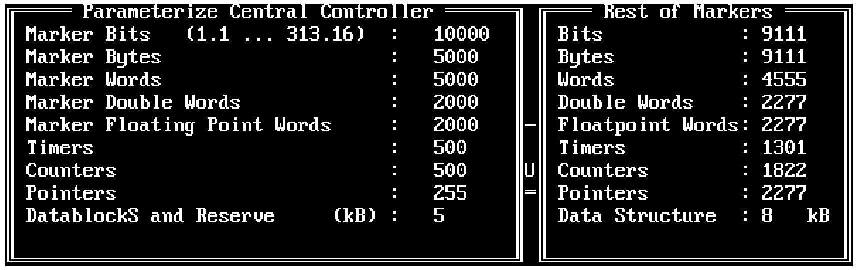
6.6.4.4 Edit menu for the equipment list editor

This menu is used mainly for configuring the secondary subrack (mounted components, addresses, timeout, etc.). You can also use it to define the segmentation of the signal memory for the controller. In the following example look at the presetting only.

Step 1 Enter <Ctrl>+<Return> to open the menu

Step 2 Enter Z for ”ParameteriZe Central Controller”

Response The following menu appears (presettings)



```
═══ Parameterize Central Controller ═══     ═══ Rest of Markers ═══
Marker Bits   (1.1 ... 313.16)  :   10000    Bits            : 9111
Marker Bytes                    :   5000     Bytes           : 9111
Marker Words                    :   5000     Words           : 4555
Marker Double Words             :   2000     Double Words    : 2277
Marker Floating Point Words     :   2000  ─  Floatpoint Words: 2277
Timers                          :   500      Timers          : 1301
Counters                        :   500   U  Counters        : 1822
Pointers                        :   255   =  Pointers        : 2277
DatablockS and Reserve     (kB) :   5        Data Structure  : 8   kB
```

Step 3 Press <Esc> to exit the menu

Response A message asks you whether you want to break off without saving; answer it with Y for ”Yes”

6.6.4.5 Terminate equipment list and save

☞ **Note:** You can bypass a menu and carry out a function by pressing <Ctrl>+<reference character>, as in the example shown in the following step.

Step 1 Press <Ctrl>+<T> (for Terminate); the equipment list is saved and you exit the equipment list editor.

Response The edit menu is displayed again

6.6.5 Assign symbols and comments (SYM/COM block)

As symbolic programming is intended, the signal symbols have to be defined. Ideally, this should happen before the program is created.

☞ **Note:** The steps will be dealt with in several sub–sections.

6.6.5.1 Assign symbols / comments for inputs / input word

Step 1	Enter `C` for ”Symbols and Comments”
Response	In answer to the message asking whether you wish to work without a backup file, confirm with Y for ”Yes”. The Symbols and Comments editor is displayed. The cursor is now positioned in the ”symbol” column, on the first character of signal I2.1.
Step 2	Enter `OFF` and <Tab>
Response	The cursor jumps to the ”Comment” column
Step 3	Enter as comment: `Off=1: all outputs off, Off=0: display` and confirm with <Tab>
Response	The entry is made in line I2.1 and the cursor is now in the ”Symbol” column, on the first character of signal I2.2.
Step 4	Enter: `LOAD` <Tab> `Load the bit string by using edge 0–>1` <Tab>
Response	The entry is made in line I2.2 and the cursor is now on the line for signal I2.3.

Step 5 Enter: `FREE` <Tab> `Free=0: freeze, Free=1: run` <Tab>

Response The entry is made in line I2.3 and the cursor is now on the line for signal I2.4.

Step 6 Enter <Ctrl>+<Return> to open the processing menu

Step 7 Enter `S` for ”Search Function”

Response A window opens in which you enter the signal you want to find

Step 8 Enter `I2.9` and <Return>

Response In the editor the cursor jumps to the ”Symbol” column of the signal you specified

Step 9 In accordance with Schritt 5 enter the following list as text for I2.9 to I2.16:

```
BIT1 <Tab> First bit of bit string <Tab>
BIT2 <Tab> Second bit of bit string <Tab>
BIT3 <Tab> Third bit of bit string <Tab>
BIT4 <Tab> Fourth bit of bit string <Tab>
BIT5 <Tab> Fifth bit of bit string <Tab>
BIT6 <Tab> Sixth bit of bit string <Tab>
BIT7 <Tab> Seventh bit of bit string <Tab>
BIT8 <Tab> Eighth bit of bit string <Tab>
```

Response The entries were made in lines I2.9 to I2.16 and the cursor is now on the line for the next signal: IW2.1

Step 10 Enter at IW2.1:
`BITSTRING` <Tab> `Bit representation online: I Status` <Tab>

Response The cursor is now on the line for the next signal: Q2.1

6.6.5.2 Insert comment lines in the SYM/COM block

Step 11 Enter <Ctrl>+<I> for ”Insert Linecomment”

Response A comment line will be inserted above the line where the cursor is positioned (in this case Q2.1).

Step 12 Enter the following text in the blank line:
`Outputs Q2.1 to Q2.16 will be used for the running light. The`

Response The line is full when ”The ” is entered. You hear a tone and the cursor jumps automatically to the next line.

Step 13 Enter <Ctrl>+<I> for ”Insert Linecomment”

Step 14 Enter the following text in the blank line:
`bit string set at I2.9 to I2.16 runs if` <Return>

Step 15 Enter <Ctrl>+<I> for ”Insert Linecomment”

Step 16 Enter the following text in the blank line:
`a 0–>1 edge is given on I2.2.`<Return>

6.6.5.3 Assign symbols for outputs

When the three comment lines have been entered, the cursor is on signal Q2.1.

Step 17 In accordance with Schritt 5 enter the following list as text for Q2.1 to Q2.16:
`RUN1` <Tab> <Tab>
`RUN2` <Tab> <Tab>
`RUN3` <Tab> <Tab>
`RUN4` <Tab> <Tab>
`RUN5` <Tab> <Tab>
`RUN6` <Tab> <Tab>
`RUN7` <Tab> <Tab>
`RUN8` <Tab> <Tab>
`RUN9` <Tab> <Tab>
`RUN10` <Tab> <Tab>
`RUN11` <Tab> <Tab>
`RUN12` <Tab> <Tab>
`RUN13` <Tab> <Tab>
`RUN14` <Tab> <Tab>
`RUN15` <Tab> <Tab>
`RUN16`

6.6.5.4 Symbols / Comment for markers / Assign marker word

Step 18 Enter <Ctrl>+<S> for ”Search Function”

Step 19 Enter `MW72` and <Return>

Response The cursor jumps to the first symbolic character of signal MW72.

Step 20 Enter at MW72:
`ROTATED` <Tab> `This word contains the rotated info`

Response The line is full on entering the o of info and the cursor jumps automatically to the next line.

Step 21 Enter <Ctrl>+<S> for ”Search Function”

Step 22 Enter `M15.1` and <Return>

Step 23 In accordance with Schritt 5 enter the following list as text for M15.1 to M15.4:
`HELP1` <Tab> `Help marker for edge detection` <Tab>
`HELP2` <Tab> `Help marker for edge detection` <Tab>
`HELP3` <Tab> `Help marker for edge detection` <Tab>
`HELP4` <Tab> `Help marker for edge detection` <Tab>

The text for flash SM173 exists already and is prescribed by the software.

Step 24 Enter <Ctrl>+<T> for ”Terminate”

Response You have now finished editing the SYM/COM block and the texts will be saved. The edit menu is displayed again.

6.6.6 Edit program (blocks)

In this chapter the user program will be entered.

6.6.6.1 Open block editor

Step 1	Enter `B` for ”Blocks”
Response	The block menu opens.
Step 2	Enter `B` for ”Block”
Step 3	Enter `PB1` for program block 1 and press <Return>
Step 4	`I` for ”Input mode”: keep toggling until ”IL” appears
Step 5	Enter `A` for ”Addressing”: keep toggling until ”SYM” appears
Step 6	Enter `S` for ”Start Entry”
Response	The block editor opens and the last network of PB1 appears on selecting ”Block end”.

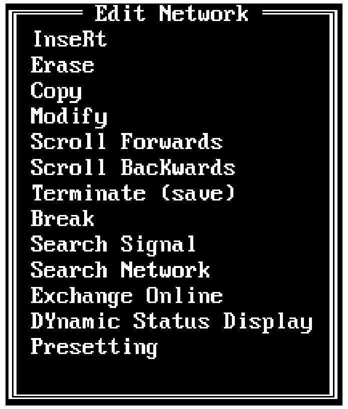
6.6.6.2 Edit PB1

First the program block (PB) with the running light program is edited. The PB1 consists of network 1 with the program and network 2 with ”BE” for block end.

❐ **Insert network**
A blank block only ever consists of network 1, which is where the block end is located. In order to edit within the block, you must first insert a blank network.

Step 1 Press <Ctrl>+<Return>

Response The processing menu opens


Edit Network
InseRt
Erase
Copy
Modify
Scroll Forwards
Scroll BacKwards
Terminate (save)
Break
Search Signal
Search Network
Exchange Online
DYnamic Status Display
Presetting

Step 2 Enter R for ”InseRt” (Network)

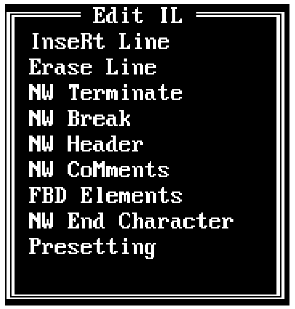
Response With the ”Insert” function, a new network is always inserted in front of the current network. In this case, network 1 is now blank (containing only :***); ”Block end” is located in network 2.

❒ Edit network 1

In network 1, the instructions for the program are now created. The program contains some jumps, which are generally edited in instruction list. Therefore the entire program will be created in instruction list, though this is not absolutely necessary.

Step 1 Press <Ctrl>+<Return>

Response The processing menu opens



```
Edit IL
InseRt Line
Erase Line
NW Terminate
NW Break
NW Header
NW CoMments
FBD Elements
NW End Character
Presetting
```

Step 2 Press `R` for ”InseRt Line”

Step 3 Now enter the following lines. When you do this, more empty lines will be created automatically during editing.

```
A <Tab> OFF <Return>
JF <Tab> =JUMP1 <Return>
L <Tab> K0 <Return>
= <Tab> ROTATED <Return>
L <Tab> K0 <Return>
JI <Tab> =JUMP3 <Return>
```

Step 4 Press <Ctrl>+<R> to insert a blank line.

Step 5 For the jump target JUMP1, move the cursor along the blank line with the left arrow key towards the edge of the screen:
`JUMP1` <Tab> (to :) `A` <Tab> `LOAD` <Return>

Step 6 Enter the following lines in accordance with Schritt 3:
EDP <Tab> HELP1 <Return>
=C <Tab> HELP2 <Return>
JF <Tab> =JUMP2 <Return>
LBW <Tab> BIT1 <Return>
DBB <Tab> CNT8 <Return>
= <Tab> ROTATED <Return>

Step 7 Enter the following line in accordance with Schritt 4 and Schritt 5:
JUMP2 <Tab> (to :) A <Tab> PULSE_7 <Return>

Step 8 Enter the following lines in accordance with Schritt 3:
A <Tab> FREE <Return>
EDP <Tab> HELP3 <Return>
=C <Tab> HELP4 <Return>
JF <Tab> =JUMP4 <Return>
A <Tab> ROTATED <Return>
ROL <Tab> V1 <Return>
= <Tab> ROTATED <Return>

Step 9 Enter the following line in accordance with Schritt 4 and Schritt 5:
JUMP3 <Tab> (to :) L <Tab> ROTATED <Return>

Step 10 Enter the following lines in accordance with Schritt 3:
TBW <Tab> RUN1 <Return>
DBB <Tab> CNT16 <Return>

Step 11 Enter the following line in accordance with Schritt 4 and Schritt 5:
JUMP4 <Tab> (to :) NOP <Return>
:***

❒ **Deleting superfluous blank instruction lines**
:*** completes the network. Redundant lines can be cleared as follows:

Step 1 Move the cursor to a blank line with the arrow keys

Step 2 Enter <Ctrl>+<E for ”Erase Line”

❒ **Close network and save**
You can **either** close the network with Schritt 1 and Schritt 2 **or** simply run Schritt 3.

Step 1 Press <Ctrl>+<Return> to open the processing menu

Step 2 Enter T for ”NW Terminate”

Step 3 Enter <Ctrl>+<T> for ”Terminate (save)”

❒ **Close block and save**
Step 1 Enter <Ctrl>+<T> for ”Terminate”

Block PB1 is now complete and the edit menu will automatically be displayed again. The PB1 must now be linked to the organization block, since this is where the ”threads” of the program are ”woven together”. Without the organization block, the program is not runnable.

6.6.6.3 Edit OB1

❐ **Open OB**

Step 1	Enter `B` for ”Block”
Step 2	Enter `OB1` for organization block 1 and press <Return>
Step 3	Enter `S` for ”Start Entry”
Response	The block editor opens and the last network of PB1 appears on selecting ”Block end”.
Step 4	Enter <Ctrl>+<R> for ”Insert” (network)
Response	With the ”Insert” function, a new network is always inserted in front of the current network. In this case, network 1 is now blank (containing only :***); ”Block end” is located in network 2.

❐ **Call PB in OB (unconditionally, i.e., the PB is called in every scan)**

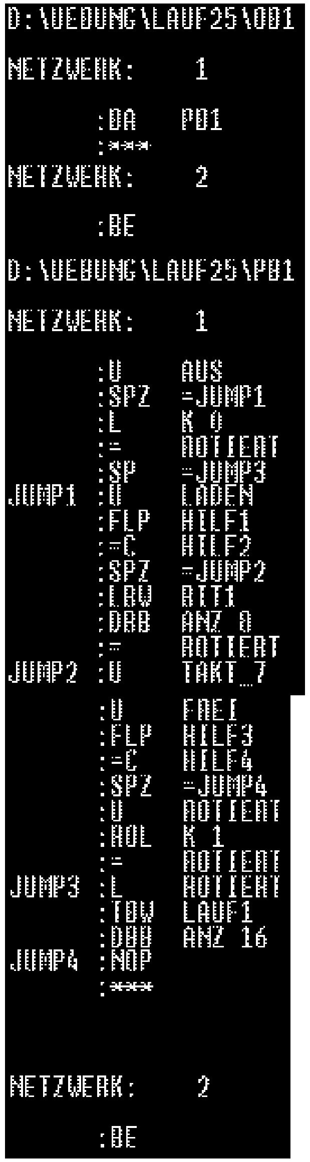
Step 1	Enter <Ctrl>+<R> for ”Insert Line”
Step 2	Enter the following text for the block call: `BC` <Tab> `PB1` <Return>

❐ **Close the OB and save**

Step 1	Enter <Ctrl>+<T> for ”Terminate” (network)
Step 2	Enter <Ctrl>+<T> for ”Terminate” (block)

With this the program input is complete.

The following shows a printout of the program which has been created.



```
D:\UEBUNG\LAUF25\OB1

NETZWERK:     1

       :BA    PB1
       :***
NETZWERK:     2

       :BE
D:\UEBUNG\LAUF25\PB1

NETZWERK:     1

       :U     AUS
       :SPZ   =JUMP1
       :L     K 0
       :=     ROTIERT
       :SP    =JUMP3
JUMP1  :U     LADEN
       :FLP   HILF1
       :=C    HILF2
       :SPZ   =JUMP2
       :LRW   BIT1
       :DRB   ANZ 0
       :=     ROTIERT
JUMP2  :U     TAKT_7
       :U     FREI
       :FLP   HILF3
       :=C    HILF4
       :SPZ   =JUMP4
       :U     ROTIERT
       :ROL   K 1
       :=     ROTIERT
JUMP3  :L     ROTIERT
       :TDW   LAUF1
       :DBB   ANZ 16
JUMP4  :NOP
       :***



NETZWERK:     2

       :BE
```

Figure 12 Your finished program should look like this

6.7 Link Program

The following function will tailor the program to PLC requirements

Step 1	To go to the main menu line, exit the open menus by pressing the <Esc> key.
Step 2	Enter L for ”Load”
Response	The load menu opens
Step 3	Enter L for ”Link Program”
Step 4	Answer the message with Y for ”Yes”
Response	The program is linked
Step 5	Confirm the message

The program is now ready to be transferred to the PLC.

6.8 Networking PLC ↔ PaDT (HW, SW)

☞ **Note:** The functions that now follow from Chapter 9.18 to Chapter 9.24 are only available with a regularly connected PLC.

Now connect the connection cable PaDT (COM1) ↔ PLC (RS 232C) and switch on the PLC power supply.

❒ Set up networking in ALD25

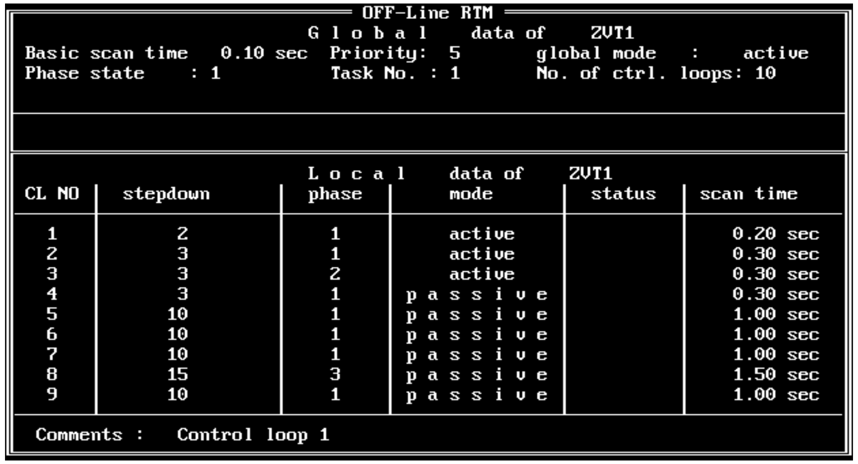
Step 1 To go to the main menu line, exit the open menus by pressing the <Esc> key.

Step 2 Enter T for "SeTup"

Response The setup menu opens

Step 3 Enter W for "Networking"

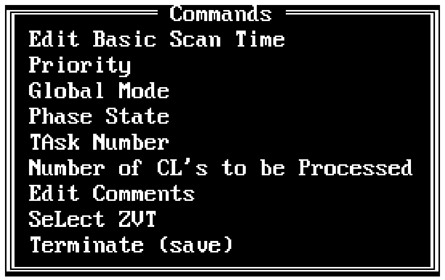
Response The connect menu opens



```
                          OFF-Line RTM
                    G l o b a l    data of    ZVT1
 Basic scan time   0.10 sec  Priority:  5      global mode   :    active
 Phase state    : 1          Task No. : 1      No. of ctrl. loops: 10


                    L o c a l    data of    ZVT1
 CL NO | stepdown        | phase |     mode      |  status  | scan time

   1   |       2         |   1   |    active     |          |  0.20 sec
   2   |       3         |   1   |    active     |          |  0.30 sec
   3   |       3         |   2   |    active     |          |  0.30 sec
   4   |       3         |   1   | p a s s i v e |          |  0.30 sec
   5   |      10         |   1   | p a s s i v e |          |  1.00 sec
   6   |      10         |   1   | p a s s i v e |          |  1.00 sec
   7   |      10         |   1   | p a s s i v e |          |  1.00 sec
   8   |      15         |   3   | p a s s i v e |          |  1.50 sec
   9   |      10         |   1   | p a s s i v e |          |  1.00 sec

  Comments :   Control loop 1
```

Step 4 Enter L for ”Local (V.24)”

Response The V.24 menu opens



```
Commands
Edit Basic Scan Time
Priority
Global Mode
Phase State
TAsk Number
Number of CL's to be Processed
Edit Comments
SeLect ZVT
Terminate (save)
```

Step 5 Enter C for ”Establish the Connection”

Response The PaDT now makes the connection, so it may be a few moments before ALD25 is ready again. When the connection has been made, a flashing double arrow appears at the bottom left corner of the screen (left / right).

☞ **Note:** If there is a program running in your PLC, switch to ”Online” by pressing <→> and run the ”StOp PLC” function

6.9 Bootload PLC

Using this function, the basic software is transferred from the PADT to the PLC and the PLC is made ready to record the program.

☞ **Note:** The following steps need only be carried out if your PLC has not yet been bootloaded with the current SW version.

Step 1	On the PLC ALU turn DIP switches B0...B2 to the right and turn DIP switch B3 to the left (take care that the PaDT ↔ PLC connecting plug does not get moved or pulled out!)
Response	The PLC is ready to transfer the basic software
Step 2	Enter L for ”Load”
Step 3	Enter B for ”Bootload”
Response	A window opens showing the transmission status. The yellow LED on the ALU keeps flashing until the transmission process has been completed error–free.
Step 4	When the bootloading is complete, confirm by pressing any key.
Step 5	On the PLC ALU turn DIP switch B3 to the right (take care that the PaDT ↔ PLC connecting plug does not get moved or pulled out!)
Response	The PLC is now ready to transfer the user program.

☞ **Note:** The green LED on the ALU should not flash during and after bootloading. If it does, the PLC must be turned off and on again and the bootloading process repeated.

6.10 Load program to PLC

Using the following function, the program, the equipment list and the initial values are transferred to the PLC.

Step 1	To go to the main menu line, exit the open menus by pressing the <Esc> key.
Step 2	Enter L for ”Load”
Response	The load menu opens
Step 3	Enter P for ”Program to PLC”
Response	The program is transferred (acknowledge completion message). The program can now be started.

6.11 Start program

Step 1	Switch to ”Online” with <→>
Response	The load menu closes and the online menu opens
Step 2	Press R for ”StaRt PLC”
Response	A message appears asking if you really want to start
Step 3	Press Y for ”Yes”
Response	The yellow LED lights up on the ALU.

6.12 Setting and changing parameters

To obtain an output on Q2.1 to Q2.16, follow the steps described below (both simulators SIM 011 are attached to I2.1 to I2.16).

Step 1	Set up any bit string (0/1) on the switches on I2.9 to I2.16.
Step 2	Set I2.1 to ”1” and the ”0” (off)
Response	After initialization (I2.1 = 1) all outputs are off
Step 3	Set I2.3 to ”1”
Response	The running light is enabled (”run”)
Step 4	Alter I2.2 from ”0” to ”1” (0 → 1 edge)
Response	The input bit string is accepted at the outputs and rotated. The program is now in its required end–state.

The bit string can be changed at any time and re–accepted by using I2.2.

6.13 Online List / Dyn. status display

6.13.1 Online List

First a list is created in which you enter the signals. In the following example the list is used for displaying the status. Signals can also be controlled or defined (forced).

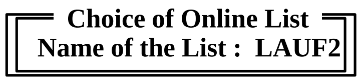
Step 1 After starting the PLC you will already be in the online menu. If not: press <Esc> to exit the open menus and then press 0 for ”Online”.

Step 2 Press L for ”Online List”

Response The das Load/Erase On–line List menu appears

Step 3 Press L for ”Load Online List”

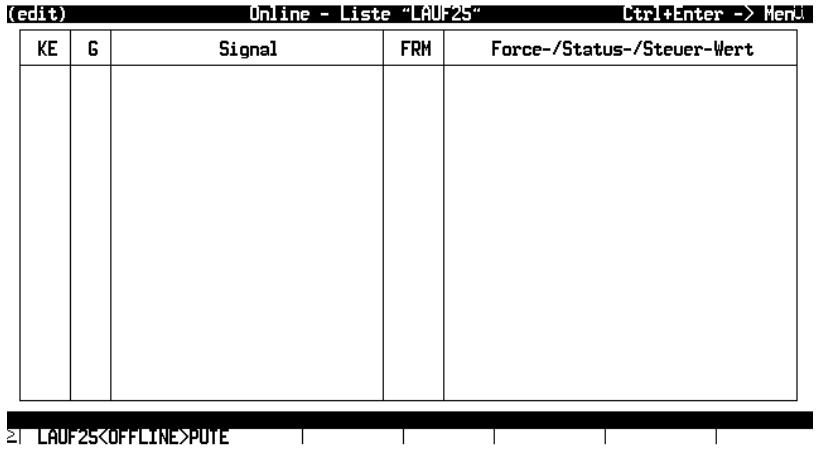
Step 4 Enter `RUN25` and press <Return> (STANDARD is automatically overwritten)



Choice of Online List

Name of the List : LAUF2

Response Confirm the message asking if you want to create the list by pressing Y for ”Yes”. The window with the list editor opens. You will find the cursor in the top, left–hand column.



```
(edit)                    Online - Liste "LAUF25"          Ctrl+Enter -> Menü
 KE | G | Signal | FRM | Force-/Status-/Steuer-Wert

≥| LAUF25<OFFLINE>PUTE
```

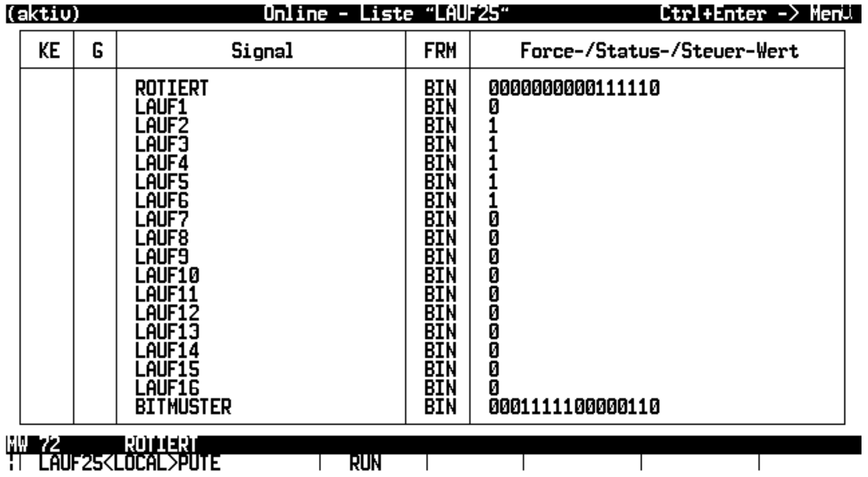
Step 5 Enter the following text in the specified sequence:
<Tab><Tab> `MW72` <Return> `B` (for binary display)
<Return>
<Tab><Tab> `Q2.1–16` <Return>
<↓> (repeat until next blank line)
<Tab><Tab> `IW2.1` <Return> `B` (for binary display)
<Return>
<Tab><Tab> `I2.1–16` <Return>

Response At each of the inputs Q2.1–16 and I2.1–16, all 16 signals are automatically declared. The list input is closed. You can now use the arrow keys or <PgUp>, <PgDn> to scroll through. (Ensure that the NUM-LOCK key is switched off.)

Step 6 Press <Ctrl>+<Return> to open the processing menu

Step 7 Enter U for "StatUs display"

Response The top line switches to "active". To continue editing, switch back to "edit" again by pressing <Esc>. The following figure shows an example of a status display using Online List.



(aktiv) Online - Liste "LAUF25" Ctrl+Enter -> Menü

KE	G	Signal	FRM	Force-/Status-/Steuer-Wert
		ROTIERT	BIN	0000000000111110
		LAUF1	BIN	0
		LAUF2	BIN	1
		LAUF3	BIN	1
		LAUF4	BIN	1
		LAUF5	BIN	1
		LAUF6	BIN	1
		LAUF7	BIN	0
		LAUF8	BIN	0
		LAUF9	BIN	0
		LAUF10	BIN	0
		LAUF11	BIN	0
		LAUF12	BIN	0
		LAUF13	BIN	0
		LAUF14	BIN	0
		LAUF15	BIN	0
		LAUF16	BIN	0
		BITMUSTER	BIN	0001111100000110

```
MW 72      ROTIERT
:| LAUF25<LOCAL>PUTE        | RUN |      |      |      |
```

Step 8 Enter <Ctrl>+<T> to close

Step 9 Answer the message asking if you want to save by pressing Y for "Yes".

Step 10 Exit the "Online List" menu by pressing <ESC>

The online list is now complete. You can now also view signal status in the dynamic status display.

6.13.2 Dynamic Status Display

Now a run–through of the dynamic status display (”current display”).

Step 1	In the online menu: enter `D` for ”Dyn. Status Display”
Response	The menu opens

```
Dynamic Status Display
Current Display
Triggered Recording
Output Mode      IL
Adressing        SYM
```

Step 2	Enter `L` for ”Current dispLay”
Step 3	Enter <↓> at ”Block” and press <Return>
Step 4	Enter `PB1` and press <Return>
Step 5	Enter `S` for ”Start”
Response	Network 1 of PB1 is selected

You can now scroll within the network using the arrow keys or scroll to the other network using <PgDn> and <PgUp>. You can press <Ctrl>+<Return> to open a processing menu in which various functions are available, e.g. viewing Online List, etc.

To view signal status scan by scan (e.g. for diagnostic purposes) you can use ”triggered recording”.

6.14 Further practise (problem–solving)

A suggestion for further practise: modifying this program. Change the program so that the bit string is no longer set up with inputs, but with help markers.

- ❐ **Select the PB1 and network 1 under ”Edit”, ”Blocks”**
- ❐ **Select ”Modify” from the processing menu**
- ❐ **Change the bit string from I2.9 to I2.16 onto M12.1 to M12.8 by entering marker M12.1 instead of BIT1**
- ❐ **Close the network and the block**
- ❐ **Transfer the resulting block online to the PLC without** linking first (”Load”, ”Exchange Online”)
- ❐ **Then modify the online list by entering M12.1 ... M12.8 instead of Bit1 ... Bit8**
- ❐ **In the ”ID” column of the online list, enter ”CE” (for control enable) by the markers**
- ❐ **Then assign the markers the required bit string in the ”Force–/Status–/Control value” column**
- ❐ **Open the processing menu and enter the values in the PLC using ”Control Enable”**
- ❐ By entering 0 → 1 edge at I2.2 the new bit string is transferred to the outputs.

6.15 Remarks about program documentation

You can use the ”Print” menu to carry out program documentation. If you use ”Entire Documentation”, all the important data are printed out in one run (complete with table of contents).

You can decide whether you want the lists to go to the screen, to a file or to the printer.

You can edit the files with any ASCII editor. You are in the current station directory. Assign the name of the file yourself when you select ”Output Unit”, ”File”.

When sending to the printer, make sure it has been initialized. Initialization is carried out in the ”SeTup”, ”Print” menu.

☞ **Note:** Please note the documentation of menus in the ”Configuration A250” user instructions.

6.16 Remarks about data security

You can use the ”Special” menu to backup or compress the entire station (operating system–independent backup) or to restore/decompress it.

☞ **Note:** Please note the documentation of menus in the ”Configuration A250” user instructions.

Chapter 7
First programming steps A120/AKF125 with ALU 204, 205

This chapter includes an example of a small application, complete in all details, for programming the A120 with the A250 feature of AKF125.

7.1 Introduction

This chapter will show you, as a beginner in AKF, how to take the first steps in programming an A120. For this, a simple program will be created in AKF125, loaded into the PLC and then viewed using dynamic status display.

7.2 Preparations

The following preparations should already have been made:

❒ On the programming unit (in this case a P810C) you will find the software installed on drive C: (see ”Installation” in the user instructions).

☞ **Note:** The sample program is installed along with the software.

❒ Included in the example is an A120 with the following modules:
Subracks DTA 205, ALU 205, DEP 216 (on slot reference 2), DAP 216 and two SIM 011s as simulators on DEP inputs I2.1 to I2.16
Please also remember the PaDT connection cable ↔ PLC.

7.3 Task definition

The idea is to create a program where an 8 bit wide bit string traverses on 16 bits of an output module (”running light”). The bit string which is going to be used is set by using inputs I2.9 to I2.16 and accepted via input I2.2. It is possible to stop the output by using I2.1 = 1 (all 16 outputs = 0) or to ”freeze” the current status by using I2.3 = 0. Programming is carried out in the special language Instruction List and symbolically.

The sample plant is called ”EXERCISE”, the sample program is called ”RUN12”.

☞ **Note:** The program logic already exists; what we have here is an exercise in handling AKF

7.4 Wiring diagram

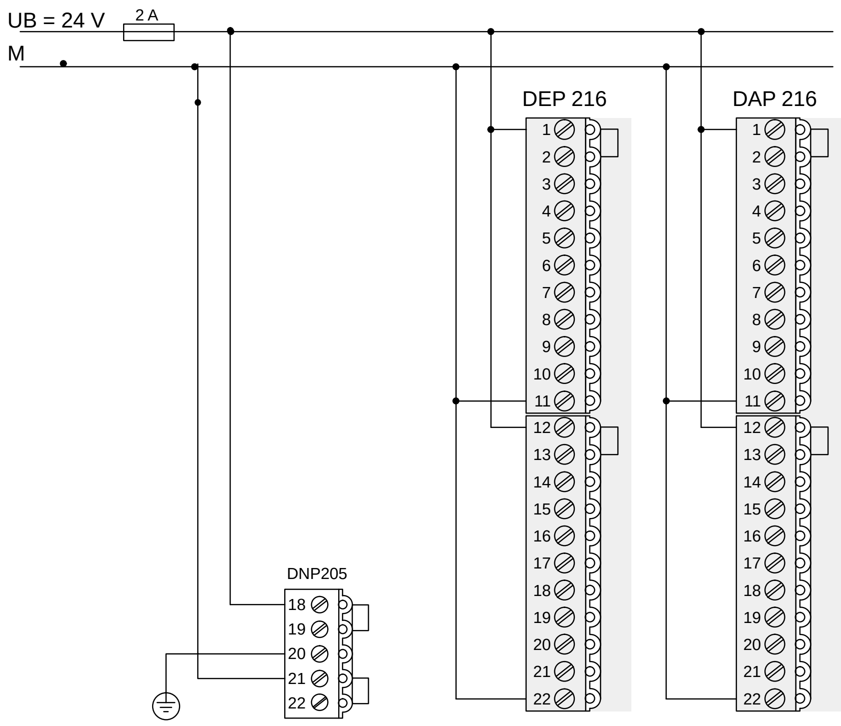


Figure 13 Hardware power supply

7.5 Parameters of the sample program

The following parameters are used in the program:

Table 3 Operands in the sample program

Operand	Symbol	Comment
I2.1	OFF	Off=1: all outputs off, Off=0: display
I2.2	LOAD	0->1 edge loads bit string
I2.3	FREE	Free=0: freeze, Free=1: run
I2.9	BIT1	First bit of bit string
I2.10	BIT2	Second bit of bit string
I2.11	BIT3	Third bit of bit string
I2.12	BIT4	Fourth bit of bit string
I2.13	BIT5	Fifth bit of bit string
I2.14	BIT6	Sixth bit of bit string
I2.15	BIT7	Seventh bit of bit string
I2.16	BIT8	Eighth bit of bit string
Q3.1	RUN1	
Q3.2	RUN2	
Q3.3	RUN3	
Q3.4	RUN4	
Q3.5	RUN5	
Q3.6	RUN6	
Q3.7	RUN7	
Q3.8	RUN8	Outputs on which the bit string
Q3.9	RUN9	is alternately sent
Q3.10	RUN10	(running light)
Q3.11	RUN11	
Q3.12	RUN12	
Q3.13	RUN13	
Q3.14	RUN14	
Q3.15	RUN15	
Q3.16	RUN16	
M15.1	HELP1	Help marker for edge detection
M15.2	HELP2	Help marker for edge detection
M15.3	HELP3	Help marker for edge detection
M15.4	HELP4	Help marker for edge detection
MW72	ROTATED	This word contains the rotated info
SM15	TAKT	10.0 Hz flashing rate

7.6 Programming

☞ **Note:** Menu functions are declared in ”inverted commas”, e.g. ”Edit”, ”Blocks”. Entries that you input (type) are in `Courier`, e.g. `AKF12`. Key combinations/special keys are shown in brackets, e.g. <Ctrl>+<S>.

7.6.1 Call program

Step 1 Call the software from user drive C: with the following command: `AKF125`

☞ **Note:** The following steps apply for the first software call after installation. If the AKF125 software has already been used for configuration and a plant has already been created, ignore the following steps and proceed as shown in chapter 11.6.2.

Response You will be asked to enter an AKF plant name.

Step 2 Enter a plant name of your choice, e.g.: `C:\PLANT`, then press <Return>

Response You will be asked whether you wish the plant to be created.

Step 3 Confirm this by pressing Y for Yes or move the menu bar to YES with the arrow keys and press <Return>.

Response You are asked to enter an AKF station name.

Step 4	Enter the station name of your choice, e.g.: `STATION`, and press <Return>
Response	You will be asked whether you wish the station to be created.
Step 5	Confirm by pressing Y for Yes or move the frame to YES with the arrow keys and press <Return>.
Response	The Dolog AKF main menu for A120 and A250 will be displayed.

7.6.2 Checking the ALU Group

☞ **Note:** The marked capitals displayed in a different color (reference characters) are for calling the menu directly.
All the steps mentioned below should be carried out in the order shown (even when the numbering begins again at ”1” in every sub–step).

Step 1	Enter T for ”SeTup”
Response	The setup menu opens
Step 2	Enter S for ”PLC station”
Step 3	Enter A for ”ALU Group”. The current entry should be ”15X; 204; 205”. If not, keep toggling until the required display is obtained.
Step 4	Press <Esc> to exit the Station menu
Response	If it has been selected again, you are asked if you really want to select this ALU group.

Step 5 Confirm by pressing Y for Yes or move the frame to YES with the arrow keys and press <Return>.
From here on we shall just refer to this as ”Confirm with Yes”.

7.6.3 Programming presettings

We will now take a look at the setups.

If the presettings are not yet as you want, carry out the following settings:

Step 1	Enter `T` for ”SeTup”.
Response	The setup menu opens.
Step 2	Enter `S` for ”PLC Station”.
Response	The Station menu opens.
Step 3	Call a new, predefined station by entering <Return>, <Blank>,<Return>.
Response	A selection window with the current AKF Stations opens.
Step 4	Select the required station with the cursor and accept by pressing <Return>.
Response	The new station is accepted into the Station menu.
Step 5	Enter `A` for ”Addressing”; keep toggling until ”SYM” appears (symbolic programming)
Step 6	`B` for ”Max. No. of Blocks”: enter `100` <Return>
Step 7	Answer the prompt with YES.
Step 8	`M` for ”Link Mode”: use the arrow keys to select ”Complete Representation” <Return>
Step 9	Select `T` for ”BusType”.
Response	Bus Type selection window opens.
Step 10	`S` for ”Modnet 1/SFB”: keep toggling until ”no” appears

Step 11 E for ”MMSE”: keep toggling until ”no” appears

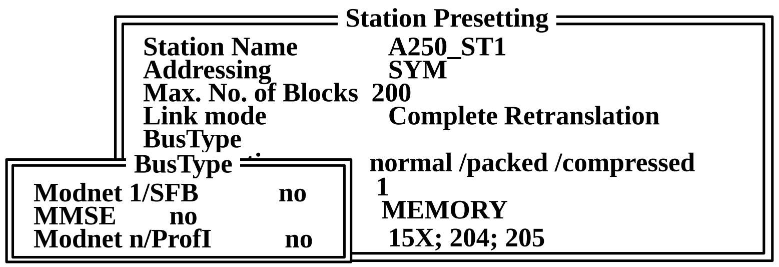
Step 12 I for ”ProfI: keep toggling until ”no” appears

Step 13 Exit the menu window with <ESC>

Step 14 Select L for Load Station. Keep toggling until the required load procedure overlays (**normal**, packed or compressed).

Step 15 Y for ”SYM–Start char.”: enter 1 <Return>

Step 16 R for ”OveRview mode”: keep toggling until ”MEMORY” appears



```
                 Station Presetting
       Station Name          A250_ST1
       Addressing            SYM
       Max. No. of Blocks  200
       Link mode             Complete Retranslation
       BusType
     BusType               normal /packed /compressed
  Modnet 1/SFB        no    1
  MMSE      no               MEMORY
  Modnet n/ProfI       no     15X; 204; 205
```

Note: Menu item ALU Group retains the setting 15X ...

Step 17 Press <Esc> twice

Response The menus close and the cursor bar stays on ”SeTup” in the main menu line. The setups have now been accepted.

7.6.4 Edit equipment list

7.6.4.1 Activate equipment list editor

Step 1 Enter E for ”Edit”

Response The edit menu opens

Step 2 Enter L for ”Equipment List”

Response The equipment list editor is displayed. A menu offers a suggestion for the first subrack.

7.6.4.2 Set up subrack

Step 1 Confirm DTA 20X by pressing <Return>

Response The first five lines are prepared for input (the grid fades out) and an ALU 205 is entered in the line for slot 0.

Step 2 If another ALU is preferred, press <Return>, selection ”ALU type”, and you can choose another ALU of type 20? by pressing <Return>.

7.6.4.3 Enter modules

Step 1 Use <↓> to move the bar to the line for slot 1

Step 2 To open the module menu, press <Return>

Step 3 Enter S for ”Special”

Response The special module menu opens

Step 4 Use <↓> to move the bar to DNP 205 and confirm with <Return>

Response DNP 205 is entered under slot 1

Step 5 Use <↓> to move the bar to the line for slot 2

Step 6 To open the module menu, press <Return>

Step 7 Enter `D` for ”Digital I/O”

Response The digital module menu opens

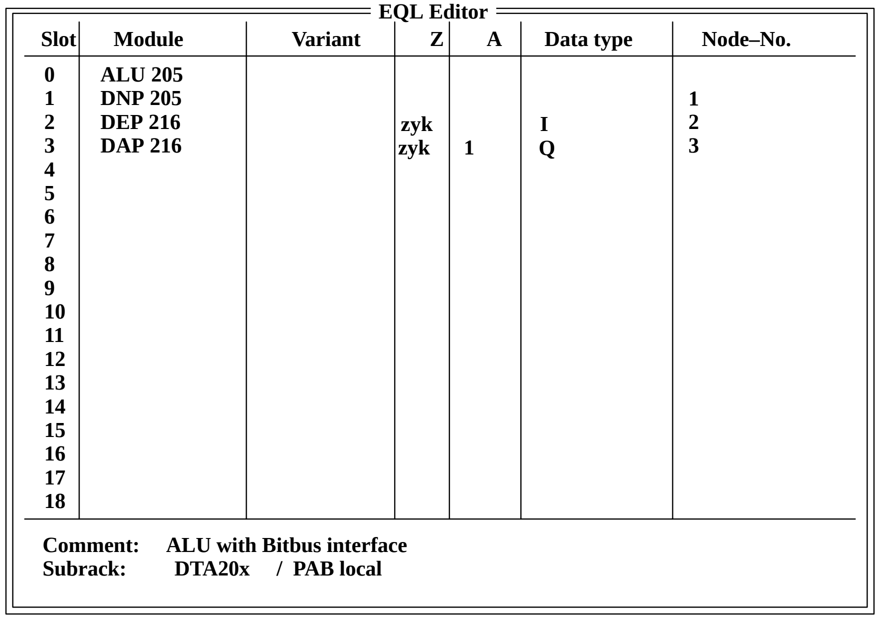
Step 8 Use <↓> to move the bar to DEP 216 and confirm with <Return>

Response DEP 216 is entered under slot 2

Step 9 Use <↓> to move the bar to DAP 216 and confirm with <Return>

Response DAP 216 is entered under slot 3

The following figure shows how your equipment list should now look.



EQL Editor

Slot	Module	Variant	Z	A	Data type	Node–No.
0	ALU 205					
1	DNP 205					1
2	DEP 216		zyk		I	2
3	DAP 216		zyk	1	Q	3
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						

Comment: ALU with Bitbus interface
Subrack: DTA20x / PAB local

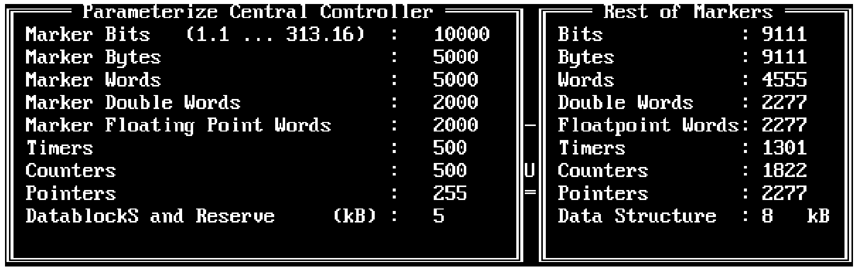
7.6.4.4 Edit menu for the equipment list editor

This menu is used mainly for configuring the secondary subrack (mounted components, addresses, timeout, etc.). You can also use it to define the segmentation of the signal memory for the controller. In the following example look at the presetting only.

Step 1 Enter <Ctrl>+<Return> to open the menu

Step 2 Enter Z for ”ParameteriZe Central Controller”

Response The following menu appears (presettings)



```
Parameterize Central Controller                      Rest of Markers
Marker Bits   (1.1 ... 313.16)  :   10000      Bits            : 9111
Marker Bytes                    :   5000       Bytes           : 9111
Marker Words                    :   5000       Words           : 4555
Marker Double Words             :   2000       Double Words    : 2277
Marker Floating Point Words     :   2000    –  Floatpoint Words: 2277
Timers                          :   500        Timers          : 1301
Counters                        :   500     U  Counters        : 1822
Pointers                        :   255     =  Pointers        : 2277
DatablockS and Reserve     (kB) :   5          Data Structure  : 8   kB
```

Step 3 Press <Esc> to exit the menu

Response A message asks you whether you want to break off without saving; answer it with Y for ”Yes”

7.6.4.5 Terminate equipment list and save

☞ **Note:** You can bypass a menu and carry out a function by pressing <Ctrl>+<reference character>, as in the example shown in the following step.

Step 1 Press <Ctrl>+<T> (for Terminate); the equipment list is saved and you exit the equipment list editor.

Response The edit menu is displayed again

7.6.5 The next steps

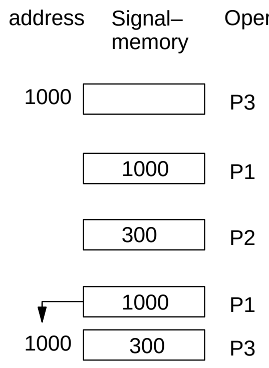
To process the program **further**, please proceed as shown in the A250 example which you will find in chapter 9. The next step is the designation of **symbol names and comments** to be found in chapter 9.16.3 beginning on page 156.

Chapter 8
Example for indirect addressing

This chapter contains several examples of indirect addressing with pointers.

8.1 Assigning a pointer indirectly to another pointer

The idea is to transfer the contents of pointer 2 indirectly (via pointer 1) to pointer 3 .



```
LA   P3   Load address of pointer 3 (1000)
          and
=    store P1        in pointer 1

L    P2   Load content of pointer 2 (300) and

=    P1–>POINTER
          store in address to which
          pointer 1 is pointing
          (address 1000 ≙ Pointer 3)
```

Pointer 2 and pointer 3 now have the same contents.

An example of a practical application of this IL segment can be found in chapter 8.10.

8.2 Assign a pointer indirectly to a data structure element

The object is to assign the contents of pointer 2 indirectly (via pointer 1) to the data structure element DB1.3.

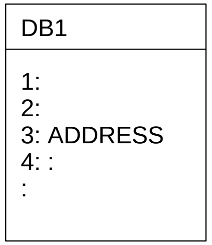


Figure 14 Organization of the data structures

```
    LA    DB1.3          Load address of DB1.3 and
    =     P1             store at pointer 1
    L     P2             Load contents of pointer 2 and
    =     P1–>POINTER    transfer to the address to which pointer 1
                         is pointing (DB1.3)
```

Pointer 2 and DB1.3 now have the same contents.

8.3 Compare a pointer indirectly with another pointer

The contents of pointer 2 are to be compared indirectly (via pointer 1) with the contents of pointer 3.

```
LA    P3            Load address of pointer 3 and
=     P1            store in pointer 1
L     P2            Load contents of pointer 2 and
>     P1–>POINTER   compare with the contents of the address to which
                    pointer 1 is pointing (contents of pointer 3)
```

8.4 Pointer as parameter in FBs

The IL segment shows the programming of a pointer as a parameter in an FB. A pointer reference (e.g. –> WORD), which is assigned to the FB as an actual operand (in this case P1), cannot be processed further within the logic of the FB. Therefore, the formal operand (in this case P) must be transferred to an additional pointer (i.e. P2) within the FB.

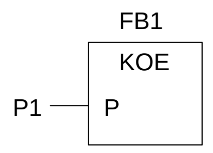


Figure 15 Pointers in FBs

A segment of the logic part of FB ”KOE”:

```
:
L       =P
=       P2
L       P2–>WORD
:
```

All other pointer processing can be carried out without reloading, e.g.:

```
:
L       P3
==      =P
:
```

8.5 Transferring a parameter from an FB to an SFB

The IL segment shows the transfer of a pointer as a parameter from an FB to an SFB.
The formal operand of FB ”KOE” cannot be transferred directly to the SFB. Therefore, the formal operand must be transferred in an additional pointer (i.e. P2).

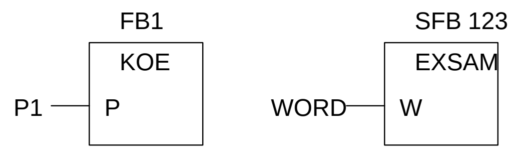


Figure 16 Pointers in FBs

A segment of the logic part of FB ”KOE”:

```
:
L       =P
=       P2
:
```

A segment of the IL of SFB ”EXAMP”:

```
BC      SFB123
W       P2–>WORD
:
```

8.6 Processing indirectly addressed values

By using indirect addressing the contents of MW100 are to be added to the contents of MW101.
In addition, at the end of the IL there is an example of a comparison with ZERO by the use of the zero pointer P_ZERO.

```
     L      K10             Load constant 10
     =      MW100           into MW100
     L      K20             Load constant 20
     =      MW101           into MW101

     LA     MW100           Load address of MW100
     =      P1              into pointer 1

     L      P1              Load pointer 1 (address of MW100)
     ADD    K2              Add constant 2
                            (address of MW100 + constant 2
                            = address of MW101) and
     =      P2              transfer into pointer 2 (now contains address
                            of MW101)

     L      P1–>WORD        Load word contents of the address to which pointer
                            1 is pointing (= contents of MW100 = 10)
     ADD    P2–>WORD        Add the word contents of the address to which
                            pointer 2 is pointing (= contents of MW101 = 20)
     =      MW102           Transfer the result (30) into MW102
     :
     :
     L      P1              Load pointer 1 (address of MW100)
     ==     P_ZERO          Compare with address 0
     JT  =L                 Jump to L if true
     :
     :
L    :
     :
     :
     ***                    IL end
```

8.7 Copying a word string from data structure DB_X to DB_Y

The aim is to copy the contents of data structure DB_X to data structure DB_Y. The number of values to be copied from DB_X to DB_Y has been defined in the first word of data structure DB_X.

```
DB_X1                      DB_Y1

1: WORD (number)           1: WORD (number)
2: WORD (value 1)          2: WORD (value 1)
3: WORD (value 2)          3: WORD (value 2)
4: WORD (value 3)          4: WORD (value 3)
:                          :
:                          :
```

Figure 17 Organization of the data structures

```
   LA    DB_X1.2       Load address of value 1 in DB_X
   =     P1            into pointer 1
   LA    DB_Y1.2       Load address of value 1 in DB_Y
   =     P2            into pointer 2
   L     DB_X1.1       Load contents of number in DB_X
   =     DB_Y1.1       into number in DB_Y
   =     MW1           and into MW1

L: L     P1–>WORD      Load word contents of the address to which pointer
                       1 is pointing (= value 1 in DB_X) and
   =     P2–>WORD      transfer word to the address to which pointer 2 is
                       pointing (= value 1 in DB_Y)

   L     MW1           Load MW1 (number of values to be copied)
   DEC                 Decrement MW1
   =     MW1           Transfer new value to MW1
   JF =E               If counting loop 0 jump to label E (end)
```

```
    L      P1            Load pointer 1 (address value 1 in DB_X)
    ADD    V2            Add constant 2 (address value 1 +
                         constant 2 = address of value 2) and
    =      P1            transfer to pointer 1. Pointer 1 is now pointing to va-
lue 2
                         in DB_X
    L      P2            Load pointer 2 (address value 1 in DB_Y)
    ADD    V2            Add constant 2 (address value 1 +
                         constant 2 = address of value 2) and
    =      P2            transfer to pointer 2 . Pointer 2 is now pointing to va-
lue 2
                         in DB_Y
    JI     =L            Jump to start of processing loop
                         (label L)

E:  ***                  IL end
```

8.8 Copying a byte string from data structure DB_X to DB_Y

The aim is to copy the contents of data structure DB_X to data structure DB_Y. The number of bytes to be copied from DB_X into DB_Y is defined in the first word of data structure DB_X. Since the signal memory consists of byte elements, it is possible to copy data structures with any type of organization by using a byte–oriented segmentation of the data–structure elements. The word ”number” in this case does not contain the number of values to be copied, but the number of bytes to be copied.

DB_X1	DB_Y1
1: WORD (number)	1: WORD (number)
2: WORD (value 1)	2: WORD (value 1)
3: BIT (value 2)	3: BIT (value 2)
4: BYTE (value 3)	4: BYTE (value 3)
5: GWORD (value 4)	5: GWORD (value 4)
:	:

Calculation of the bytes to be copied:

WORD (value 1)	+ BIT (value 2)	+ BYTE (value 3)	+ GWORD (value 4)	= Number of bytes
2+ byte	+ 1 Byte	+ 1 Byte	+ 4 Bytes	= 9 Bytes

Figure 18 Organization of the data structures

```
    LA    DB_X1.2       Load address of value 1 (low byte) in DB_X
    =     P1            into pointer 1
    LA    DB_Y1.2       Load address of value 1 (low byte) in DB_Y
    =     P2            into pointer 2
    L     DB_X1.1       Load contents of number in DB_X
    =     DB_Y1.1       into number in DB_Y
    =     MW1           and into MW1

L:  L     P1–>BYTE      Load byte contents of the address to which pointer
                        1 is pointing (= low byte of value 1 in DB_X) and
    =     P2–>BYTE      transfer byte to the address to which pointer 2 is
                        pointing (= low byte of value 1 in DB_Y)

    L     MW1           Load MW1 (number of values to be copied)
    DEC                 Decrement MW1
    =     MW1           Store new value in MW1
    JF =E               If counting loop 0 jump to label E (end)

    L     P1            Load pointer 1 (address of value 1 (low byte)
                        in DB_X)
    ADD   K1            Add constant 1 (address of low byte
                        (value 1)+ constant 1 = address of high byte
                        (value 1)) and
    =     P1            transfer to pointer 1. Pointer 1 is now pointing to
                        high byte of value 1 in DB_X

    L     P2            Load pointer 2 (address of value 1 (low byte)
                        in DB_Y)
    ADD   K1            Add constant 1 (address of low byte
                        (value 1) + constant 1 = address of high byte
                        (value 1) and
    =     P2            transfer to pointer 2. Pointer 2 is now pointing to
                        high byte of value 1 in DB_Y
    JI    =L            Jump to start of processing loop
                        (label L)

E:  ***                 IL end
```

8.9 Copying a word string from data structures with the aid of the data type address

Using the data type address, the contents of data structure DB_X can be copied to data structure DB_Y . The data type address is (like the pointer) also a pointer in its own right. It is pointing in this example to the address of value 1.

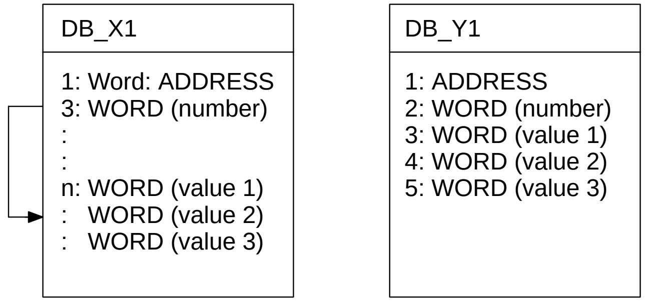


Figure 19 Organization of the data structures

```
     L      DB_X1.2      Load the contents of ADDRESS (address of value 1)
                         in DB_X
     =      P1           into pointer 1
     LA     DB_Y1.3      Load the address of value 1 in DB_Y
     =      P2           into pointer 2
     L      DB_X1.3      Load contents of number in DB_X
     =      DB_Y1.2      into number in DB_Y and
     =      MW1          into MW1

L:   L      P1–>WORD     Load word contents of the address to which pointer
                         1 is pointing (= value 1 in DB_X) and
     =      P2–>WORD     transfer the word to the address to which pointer 2 is
                         pointing (= value 1 in DB_Y)

     L      MW1          Load MW1 (number of values to be copied)
     DEC                 Decrement MW1
     =      MW1          Store new value in MW1
     JF =E               If counting loop 0 jump to label E (end)

     L      P1           Load pointer 1 (address of value 1 in DB_X)
     ADD    K2           Add constant 2 (address value 1 +
                         constant 2 = address of value 2) and
     =      P1           transfer to pointer 1. Pointer 1 now points to value 2
                         in DB_X

     L      P2           Load pointer 2 (address of value 1 in DB_Y)
     ADD    K2           Add constant 2 (address of value 1 +
                         constant 2 = address of value 2) and
     =      P2           transfer to pointer 2. Pointer 2 is now pointing to
                         value 2
                         in DB_Y
     JI     =L           Jump to start of processing loop
                         (label L)

E:   ***                 IL end
```

8.10 Copying a word string from data structures with the aid of a data structure

With the aid of ”directory block” GEDB, the contents of data structure ROLL1 can be copied into data structure UEDB1.

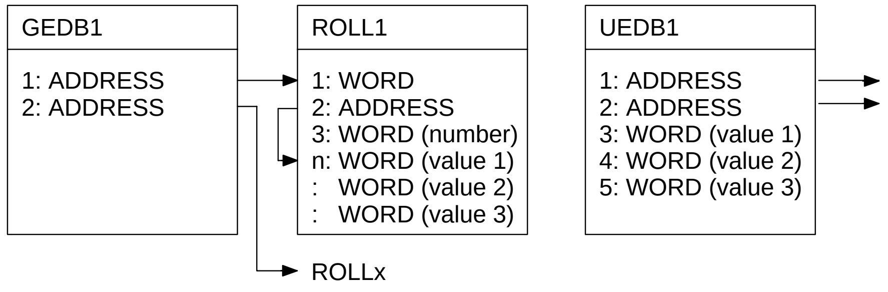


Figure 20 Organization of the data structures

```
L      GEDB1.1        Load the contents of the first ADDRESS in GEDB1
                      (address of the first Word in ROLL1)
ADD    V2             Add constant 2 and
=      P1             transfer to pointer 1. Pointer 1 is now pointing to
                      ROLL1.2
ADD    V4             Add constant 4 and
=      P2             transfer to pointer 2. Pointer 2 is now pointing to
                      ROLL1.3
L      P1–>POINTER    Load pointer (address) contents of address to which
                      pointer is pointing (address of value 1 in ROLL1)
                      and
=      P1             transfer to pointer 1. Pointer 1 is now pointing to
                      ROLL1.n
L      P2–>WORD       Load word contents of the address to which pointer
                      is pointing (address of ROLL1.3 (number)) and
=      MW1            transfer to MW1
LA     UEDB1.3        Load address of UEDB1.3 (value 1) and
=      P3             transfer to pointer 3
```

```
L:  L      P1–>WORD     Load word contents of the address to which pointer
                        1 is pointing (= value 1 in ROLL1) and
    =      P3–>WORD     Transfer word to the address to which pointer 3 is
                        pointing (= value 1 in UEDB1)

    L      MW1          Load MW1 (number of values to be copied)
    DEC                 Decrement MW1
    =      MW1          Store new value in MW1
    JF  =E              If counting loop 0 jump to label E (end)

    L      P1           Load pointer 1 (address value 1 in ROLL1)
    ADD    K2           Add constant 2 (address value 1 +
                        constant 2 = address of value 2) and
    =      P1           store in pointer 1. Pointer 1 is now pointing to value
2
                        in ROLL1

    L      P3           Load pointer 3 (address value 1 in UEDB1)
    ADD    V2           Add constant 2 (address value 1 +
                        constant 2 = address of value 2) and
    =      P3           store in pointer 3. Pointer 3 is now pointing to value
2
                        in UEDB1
    JI     =L           Jump to start of processing loop
                        (label L)

E:  ***                 IL end
```

8.11 Indirect addressing of data structures

The ”directory block” VZBS1 enables the execution of multiple data structures within one loop.

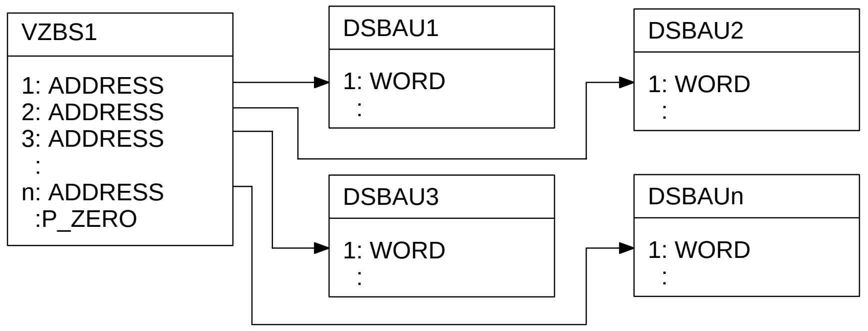


Figure 21 Organization of the data structures

```
   LA    VZBS1.1       Load address of VZBS1.1 and
   =     P1            transfer to pointer 1

A: L     P1–>POINTER   Load pointer (address) contents of the address to
                       which
                       the pointer is pointing (address of DSBAU1.1)
   ==    P_ZERO        Compare with address 0
   JT    E             Jump to E if true

   L     P1–>...       Processing
   :                   the data structure

   L     P1            Load pointer 1 (address of VZBS1.1)
   ADD   K4            Add constant 4 and
   =     P1            store result in pointer 1. Pointer is now pointing to
                       VZBS1.2
   JI    A             Jump to A

E: ***                 IL end
```

128 Example for indirect addressing

Index